London Machine Learning Meetup
20 Feb 2019

How to Combine Neural Networks and Decision Trees

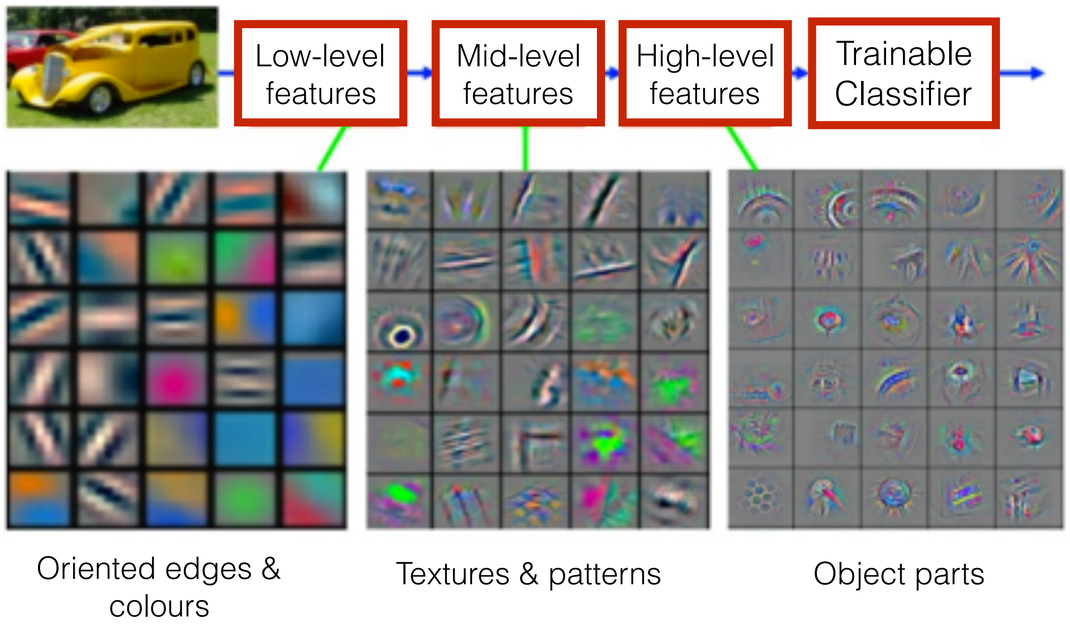
Ryutaro Tanno
University College London, UK

Two Paradigms of Machine Learning

Deep Neural Networks

『hierarchical *representation* of data』

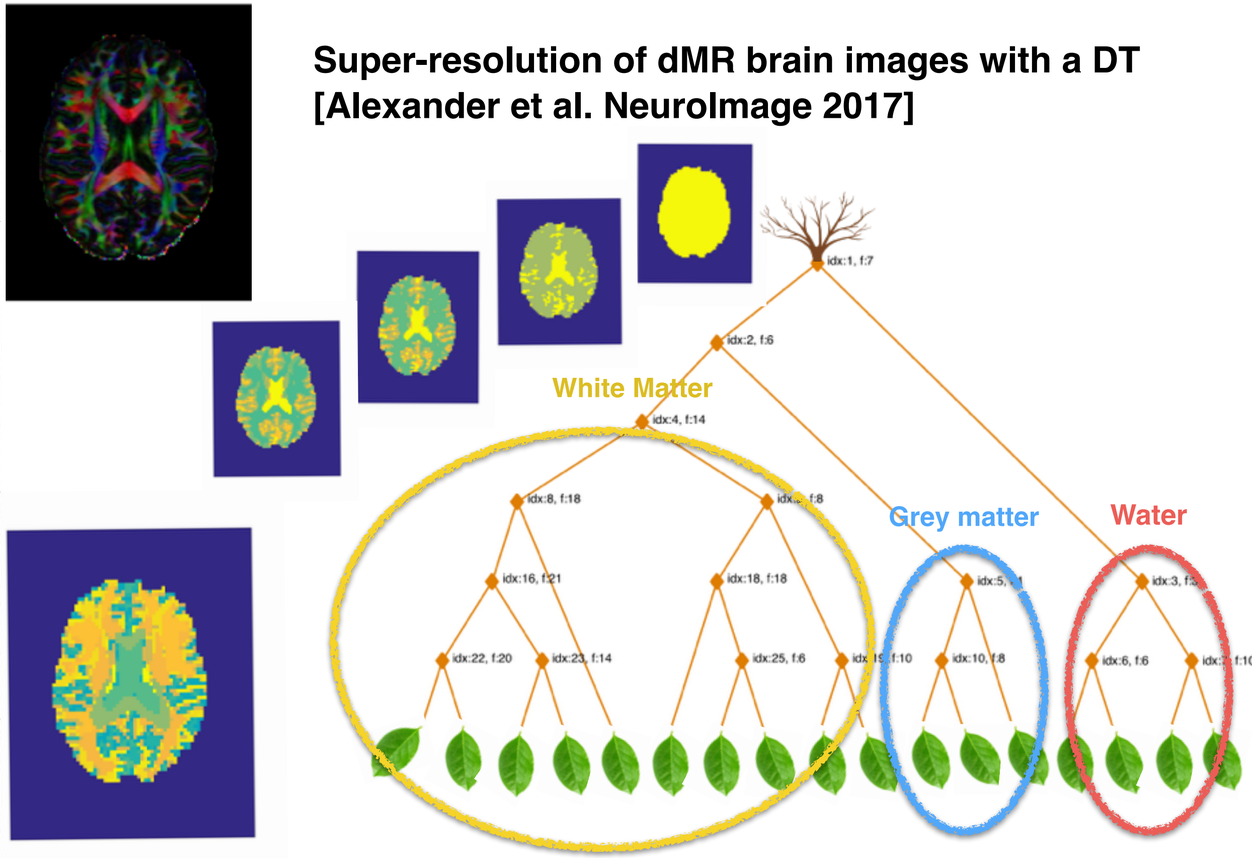
ImageNet classifiers with CNNs
[Zeiler and Fergus, ECCV 2014]



Decision Trees

『hierarchical *clustering* of data』

Super-resolution of dMR brain images with a DT
[Alexander et al. NeuroImage 2017]



Two Paradigms of Machine Learning

Deep Neural Networks

『hierarchical *representation* of data』

Decision Trees

『hierarchical *clustering* of data』

Two Paradigms of Machine Learning

Deep Neural Networks

『hierarchical *representation* of data』

\+ learn features of data

\+ scalable learning with stochastic optimisation

\- architectures are hand-designed

\- heavy-weight inference, engaging every parameter of the model for each input

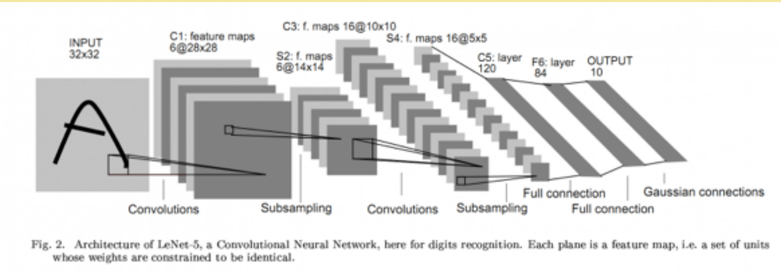


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Decision Trees

『hierarchical *clustering* of data』

Two Paradigms of Machine Learning

Deep Neural Networks

『hierarchical *representation* of data』

+ learn features of data

+ scalable learning with stochastic optimisation

- architectures are hand-designed

- heavy-weight inference, engaging every parameter of the model for each input

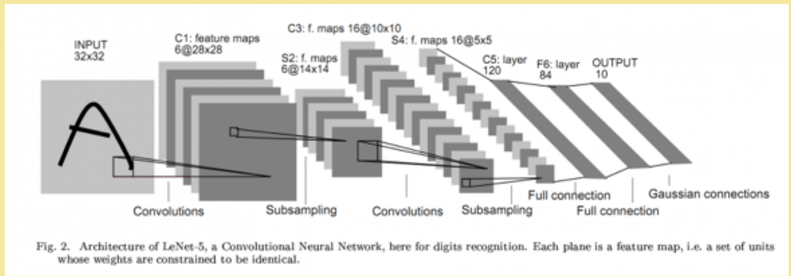


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Decision Trees

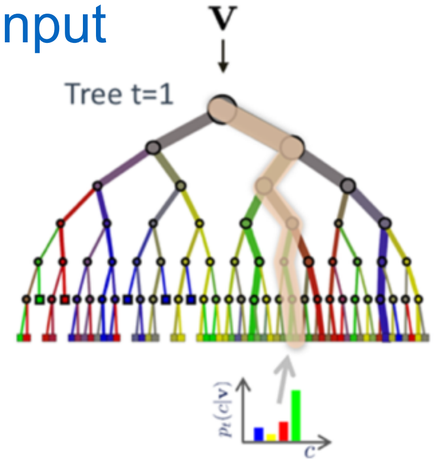
『hierarchical *clustering* of data』

- operate on hand-designed features

- limited expressivity with simple splitting functions

+ architectures are learned from data

+ lightweight inference, activating only a fraction of the model per input



Goal

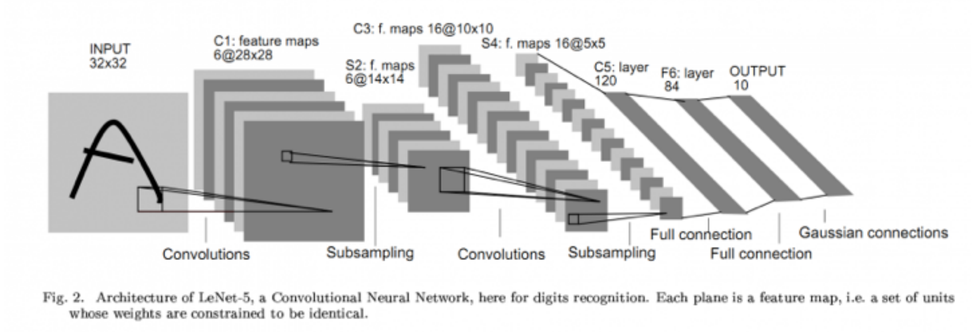
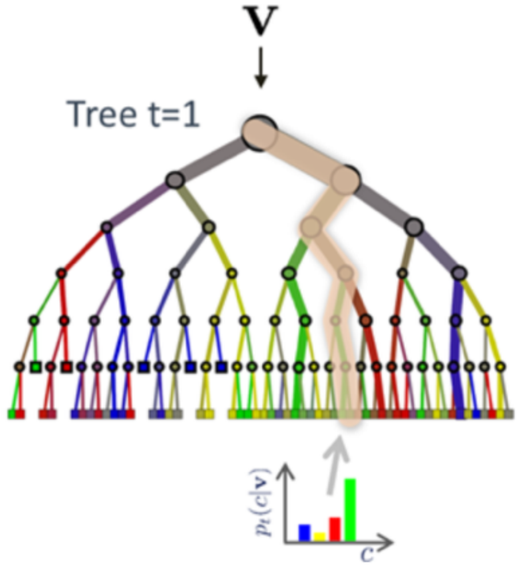


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

+



Our solution: ANTs (Adaptive Neural Trees)

Goal

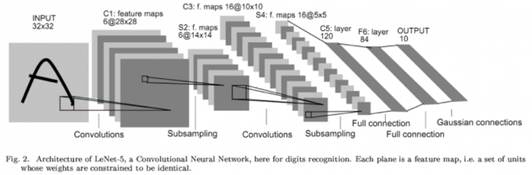
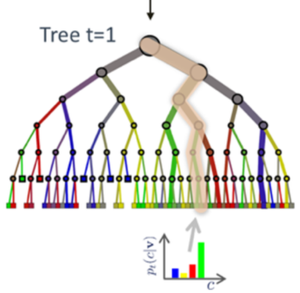


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

+



- **ANTs enjoy the benefits of the two worlds:**

Deep Neural Networks

+ feature learning
+ scalable optimisation with SGD

Decision Trees

+ architecture learning
+ lightweight inference via conditional computation

- **ANTs for supervised learning & demonstrate on regression and classification.**

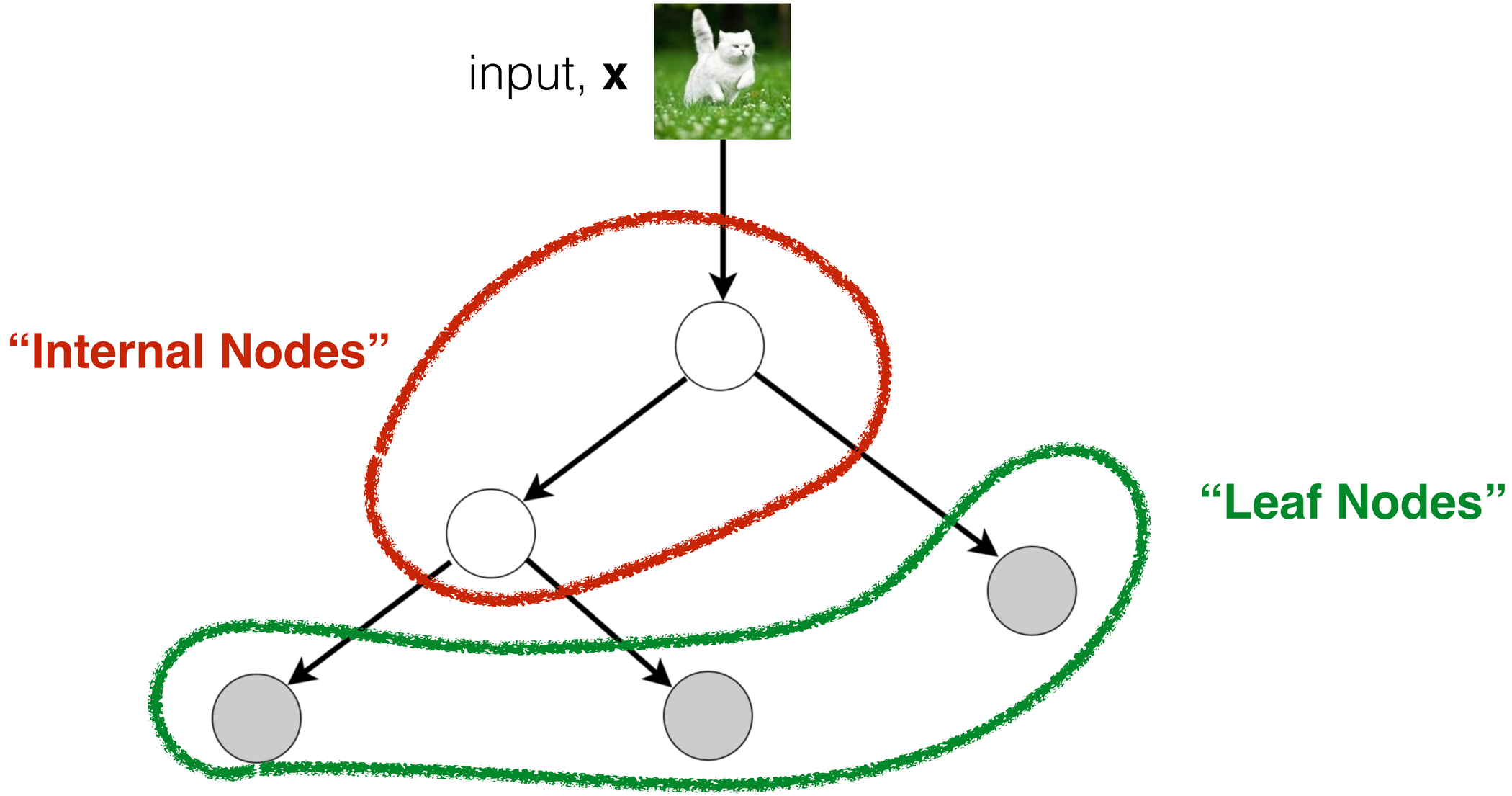
- **ANTs consist of two key designs:**

(i). DTs which use NNs in every path and routing decisions.

(ii). Back-propagation based training algorithm which grows the architecture.

Definition of ANTs

- *A **binary tree** with a set of operations*:



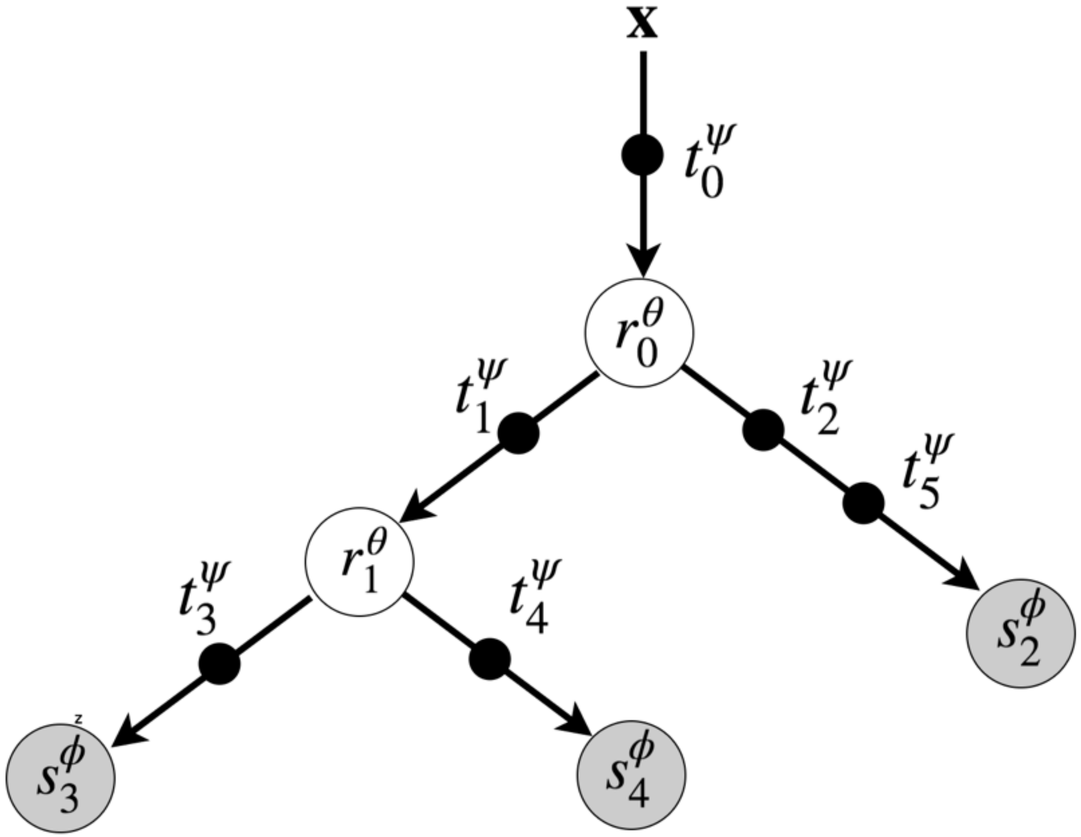
Definition of ANTs

- *A **binary tree** with **3 types** of operations*:

internal node = a ***router*** (white circles), $r_j^\theta : \mathcal{X}_j \to [0,1]$ e.g. a small CNN

edge = one or multiple **transformer** (black circles) t^ψ, e.g. 1 Conv. + ReLU

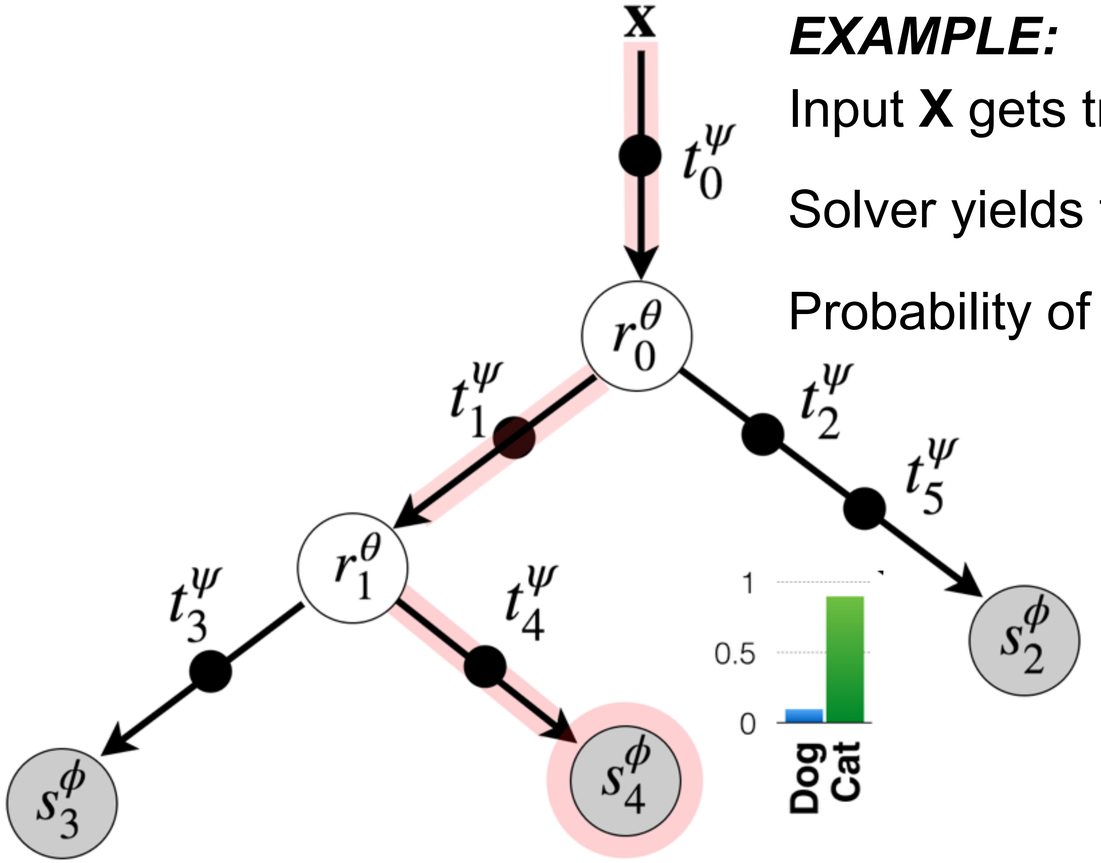
leaf node = a ***solver*** (grey circles), $s_l^\phi : \mathcal{X}_l \to \mathcal{Y}$, e.g. a linear classifier



Definition of ANTs

- *A **binary tree** with **3 types** of operations*:

internal node	=	a ***router*** (white circles), $r_j^\theta : \mathcal{X}_j \to [0,1]$	e.g. a small CNN
edge	=	one or multiple **transformer** (black circles) t^ψ,	e.g. 1 Conv. + ReLU
leaf node	=	a ***solver*** (grey circles), $s_l^\phi : \mathcal{X}_l \to \mathcal{Y}$,	e.g. a linear classifier



EXAMPLE:

Input **X** gets transformed: $\mathbf{x} \to \mathbf{x}_0^{\boldsymbol{\psi}} := t_0^{\boldsymbol{\psi}}(\mathbf{x}) \to \mathbf{x}_1^{\boldsymbol{\psi}} := t_1^{\boldsymbol{\psi}}(\mathbf{x}_0^{\boldsymbol{\psi}}) \to \mathbf{x}_4^{\boldsymbol{\psi}} := t_4^{\boldsymbol{\psi}}(\mathbf{x}_1^{\boldsymbol{\psi}})$

Solver yields the prediction: $p_4^{\boldsymbol{\phi},\boldsymbol{\psi}}(\mathbf{y}) := s_4^{\boldsymbol{\phi}}(\mathbf{x}_4^{\boldsymbol{\psi}})$

Probability of taking this path is: $\pi_4^{\boldsymbol{\psi},\boldsymbol{\theta}}(\mathbf{x}) := r_0^{\boldsymbol{\theta}}(\mathbf{x}) \cdot (1 - r_1^{\boldsymbol{\theta}}(\mathbf{x}_1^{\boldsymbol{\psi}}))$

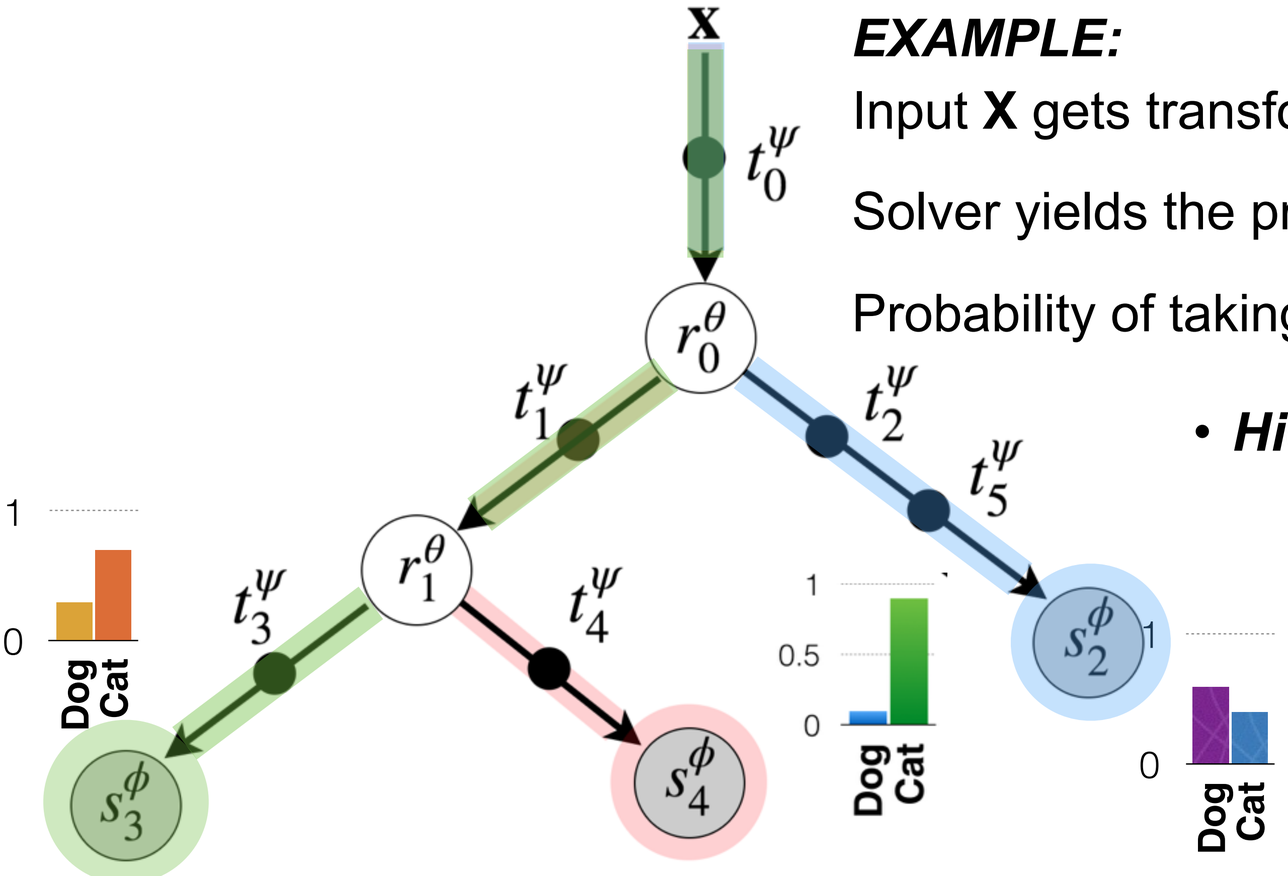
Definition of ANTs

- *A **binary tree** with **3 types** of operations*:

internal node = a ***router*** (white circles), $r_j^\theta : \mathcal{X}_j \to [0,1]$ e.g. a small CNN

edge = one or multiple **transformer** (black circles) t^ψ, e.g. 1 Conv. + ReLU

leaf node = a ***solver*** (grey circles), $s_l^\phi : \mathcal{X}_l \to \mathcal{Y}$, e.g. a linear classifier



EXAMPLE:

Input **X** gets transformed: $\mathbf{x} \to \mathbf{x}_0^\psi := t_0^\psi(\mathbf{x}) \to \mathbf{x}_1^\psi := t_1^\psi(\mathbf{x}_0^\psi) \to \mathbf{x}_4^\psi := t_4^\psi(\mathbf{x}_1^\psi)$

Solver yields the prediction: $p_4^{\phi,\psi}(\mathbf{y}) := s_4^\phi(\mathbf{x}_4^\psi)$

Probability of taking this path is: $\pi_4^{\psi,\theta}(\mathbf{x}) := r_0^\theta(\mathbf{x}) \cdot (1 - r_1^\theta(\mathbf{x}_1^\psi))$

- ***Hierarchical Mixture of Experts*** *(Jordan 1994)*

$$p(\mathbf{y}|\mathbf{x},\theta,\phi,\psi) = \sum_{l\in\{2,3,4\}} \pi_l^{\psi,\theta}(\mathbf{x}) \cdot p_l^{\phi,\theta}(\mathbf{x})$$

Full predictive distribution | **Reaching probability** | **Leaf prediction**

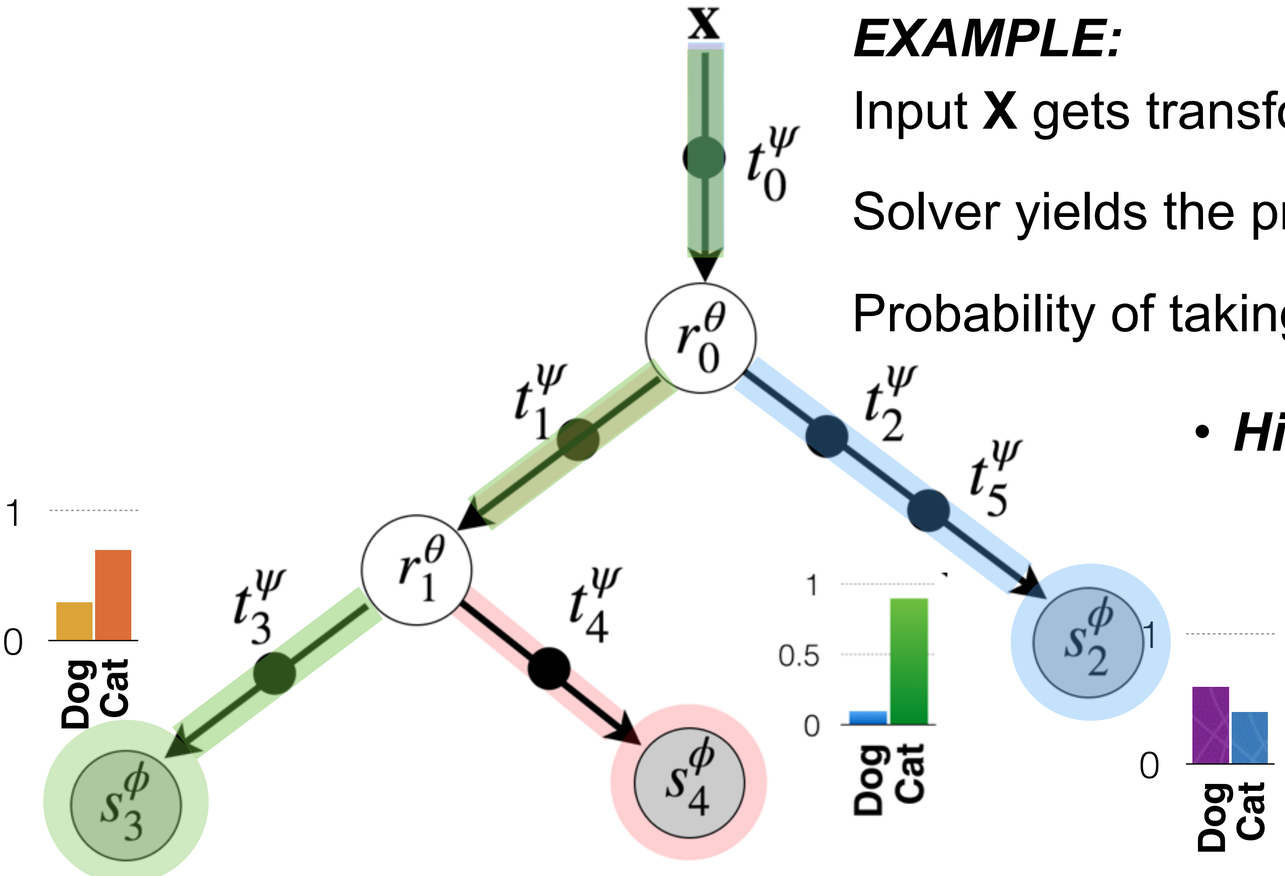
Definition of ANTs

- *A **binary tree** with **3 types** of operations*:

internal node = a ***router*** (white circles), $r_j^\theta : \mathcal{X}_j \to [0,1]$ e.g. a small CNN

edge = one or multiple **transformer** (black circles) t^ψ, e.g. 1 Conv. + ReLU

leaf node = a ***solver*** (grey circles), $s_l^\phi : \mathcal{X}_l \to \mathcal{Y}$, e.g. a linear classifier



EXAMPLE:

Input **X** gets transformed: $\mathbf{x} \to \mathbf{x}_0^\psi := t_0^\psi(\mathbf{x}) \to \mathbf{x}_1^\psi := t_1^\psi(\mathbf{x}_0^\psi) \to \mathbf{x}_4^\psi := t_4^\psi(\mathbf{x}_1^\psi)$

Solver yields the prediction: $p_4^{\phi,\psi}(\mathbf{y}) := s_4^\phi(\mathbf{x}_4^\psi)$

Probability of taking this path is: $\pi_4^{\psi,\theta}(\mathbf{x}) := r_0^\theta(\mathbf{x}) \cdot (1 - r_1^\theta(\mathbf{x}_1^\psi))$

- ***Hierarchical Mixture of Experts*** *(Jordan 1994)*

$$p(\mathbf{y}|\mathbf{x},\theta,\phi,\psi) = \sum_{l\in\{2,3,4\}} \pi_l^{\psi,\theta}(\mathbf{x}) \cdot p_l^{\phi,\theta}(\mathbf{x})$$

Full predictive distribution | **Reaching probability** | **Leaf prediction**

- ***Two inference schemes:***

(1) Multi-path & (2) Single-path

Optimisation of ANTs

- **Training of an ANT proceeds in two phases:**
  1). *Growth*: grow the architecture based on *local optimisation.*
  2) *Refinement*: fine-tunes based on *global optimisation.*

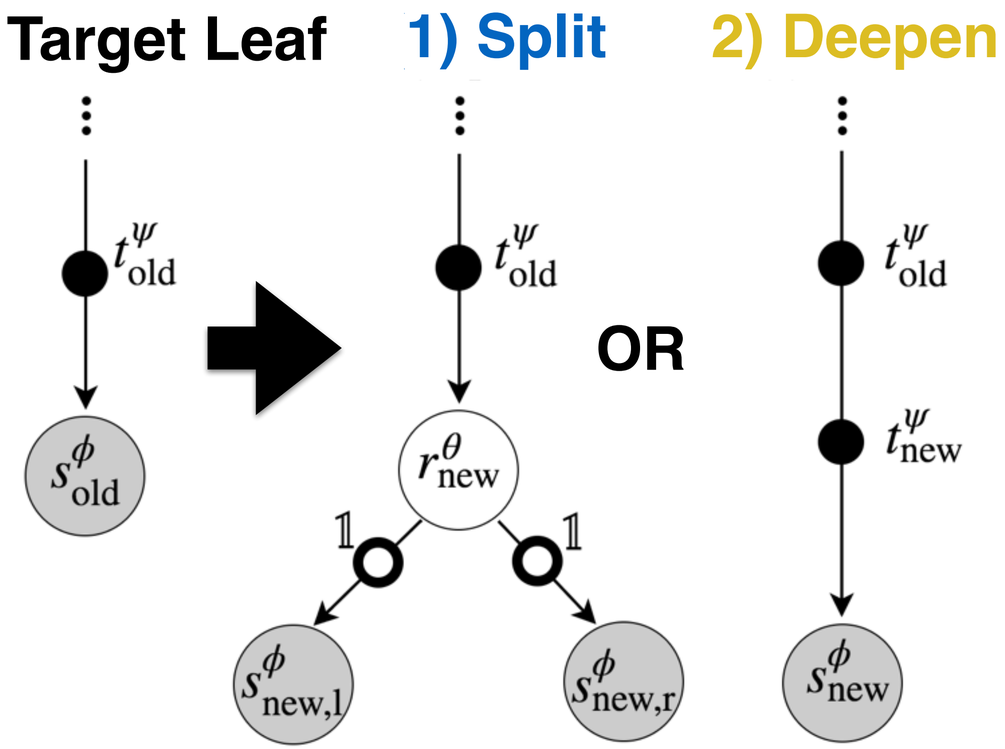
Optimisation of ANTs

- **Training of an ANT proceeds in two phases:**
  1). *Growth*: grow the architecture based on *local optimisation.*
  2) *Refinement*: fine-tunes based on *global optimisation.*

- **Objective function:** negative log likelihood (NLL)

$$-\log\, p(\mathbf{Y}|\mathbf{X},\boldsymbol{\theta},\boldsymbol{\psi},\boldsymbol{\phi}) = -\sum_{n=1}^{N}\log\Big(\sum_{l=1}^{L}\pi_l^{\boldsymbol{\theta},\boldsymbol{\psi}}(\mathbf{x}^{(n)})\;\cdot p_l^{\boldsymbol{\phi},\boldsymbol{\psi}}(\mathbf{y}^{(n)}|\mathbf{x}^{(n)})\Big)$$

Optimisation of ANTs



- **Training of an ANT proceeds in two phases:**
  1). *Growth*: grow the architecture based on *local optimisation.*
  2) *Refinement*: fine-tunes based on *global optimisation.*

- **Objective function:** negative log likelihood (NLL)

$$-\log\, p(\mathbf{Y}|\mathbf{X},\boldsymbol{\theta},\boldsymbol{\psi},\boldsymbol{\phi}) = -\sum_{n=1}^{N}\log\Big(\sum_{l=1}^{L}\pi_l^{\boldsymbol{\theta},\boldsymbol{\psi}}(\mathbf{x}^{(n)})\cdot p_l^{\boldsymbol{\phi},\boldsymbol{\psi}}(\mathbf{y}^{(n)}|\mathbf{x}^{(n)})\Big)$$

- **Details of growth phase:** at each leaf node,

(I). construct **2** models: one with a new router (**Split**) and the other with a new transformer (**Deepen**)

(II). optimise new modules (locally) by minimising NLL

(III). pick the best choice

Repeat until no more **"Split"** or **"Deepen"** pass the test.

Generalisation of previous work

	Feature learning?		Adaptive architecture?
	Routers	Path	
Soft DT, (Suarez & Lutsko, 1999)			
Soft DT 2 / HMEs (Jordan & Jacobs, 1994)			
Soft DT 3, (Frosst & Hinton, 2017)			
Convolutional DT, (Laptev & Buhman, 2014)			
Neural Decision Tree (Kontschieder et al., 2014)			
Neural Decision Forest (Kontschieder et al., 2015)			
Conditonal Net (Ioanou et al., 2016)			
ANT (Ours)			

Generalisation of previous work

	Feature learning?		Adaptive architecture?
	Routers	Path	
Soft DT, (Suarez & Lutsko, 1999)	✗	✗	✓
Soft DT 2 / HMEs (Jordan & Jacobs, 1994)			
Soft DT 3, (Frosst & Hinton, 2017)			
Convolutional DT, (Laptev & Buhman, 2014)			
Neural Decision Tree (Kontschieder et al., 2014)			
Neural Decision Forest (Kontschieder et al., 2015)			
Conditonal Net (Ioanou et al., 2016)			
ANT (Ours)			

Routers: axis-aligned features
Transformers: Identity
No representation learning ✗

Generalisation of previous work

	Feature learning?		Adaptive architecture?
	Routers	Path	
Soft DT, (Suarez & Lutsko, 1999)	✗	✗	✓
Soft DT 2 / HMEs (Jordan & Jacobs, 1994)	✓	✗	✗
Soft DT 3, (Frosst & Hinton, 2017)	✓	✗	✗
Convolutional DT, (Laptev & Buhman, 2014)	✓	✗	✗
Neural Decision Tree (Kontschieder et al., 2014)	✓	✗	✓
Neural Decision Forest (Kontschieder et al., 2015)			
Conditonal Net (Ioanou et al., 2016)			
ANT (Ours)			

Routers: axis-aligned features
Transformers: Identity
No representation learning ✗

Routers: Linear classifier, MLPs, Conv. + Linear classifier
Transformers: Identity
No representation learning ✗

Generalisation of previous work

	Feature learning?		Adaptive architecture?
	Routers	Path	
Soft DT, (Suarez & Lutsko, 1999)	✗	✗	✓
Soft DT 2 / HMEs (Jordan & Jacobs, 1994)	✓	✗	✗
Soft DT 3, (Frosst & Hinton, 2017)	✓	✗	✗
Convolutional DT, (Laptev & Buhman, 2014)	✓	✗	✗
Neural Decision Tree (Kontschieder et al., 2014)	✓	✗	✓
Neural Decision Forest (Kontschieder et al., 2015)	✓	✓	✗
Conditonal Net (Ioanou et al., 2016)			
ANT (Ours)			

Routers: axis-aligned features
Transformers: Identity
No representation learning ✗

Routers: Linear classifier
Transformers: GoogLeNet at the root edge & identity at the remaining edges
No architecture learning ✗

Routers: Linear classifier, MLPs, Conv. + Linear classifier
Transformers: Identity
No representation learning ✗

Generalisation of previous work

	Feature learning?		Adaptive architecture?
	Routers	Path	
Soft DT, (Suarez & Lutsko, 1999)	✗	✗	✓
Soft DT 2 / HMEs (Jordan & Jacobs, 1994)	✓	✗	✗
Soft DT 3, (Frosst & Hinton, 2017)	✓	✗	✗
Convolutional DT, (Laptev & Buhman, 2014)	✓	✗	✗
Neural Decision Tree (Kontschieder et al., 2014)	✓	✗	✓
Neural Decision Forest (Kontschieder et al., 2015)	✓	✓	✗
Conditonal Net (Ioanou et al., 2016)	✓	✓	✗
ANT (Ours)			

Routers: axis-aligned features
Transformers: Identity
No representation learning ✗

Routers: Linear classifier, MLPs, Conv. + Linear classifier
Transformers: Identity
No representation learning ✗

Routers: Linear classifier
Transformers: GoogLeNet at the root edge & identity at the remaining edges
No architecture learning ✗

Routers: MLPs
Transformers: combination of convolutions + ReLU + pooling
Note: this work considers Directed Acyclic Graphs.
No architecture learning ✗

Generalisation of previous work

	Feature learning?		Adaptive architecture?
	Routers	Path	
Soft DT, (Suarez & Lutsko, 1999)	✗	✗	✓
Soft DT 2 / HMEs (Jordan & Jacobs, 1994)	✓	✗	✗
Soft DT 3, (Frosst & Hinton, 2017)	✓	✗	✗
Convolutional DT, (Laptev & Buhman, 2014)	✓	✗	✗
Neural Decision Tree (Kontschieder et al., 2014)	✓	✗	✓
Neural Decision Forest (Kontschieder et al., 2015)	✓	✓	✗
Conditonal Net (Ioanou et al., 2016)	✓	✓	✗
ANT (Ours)	✓	✓	✓

Routers: axis-aligned features
Transformers: Identity
No representation learning ✗

Routers: Linear classifier, MLPs, Conv. + Linear classifier
Transformers: Identity
No representation learning ✗

Routers: Linear classifier
Transformers: GoogLeNet at the root edge & identity at the remaining edges
No architecture learning ✗

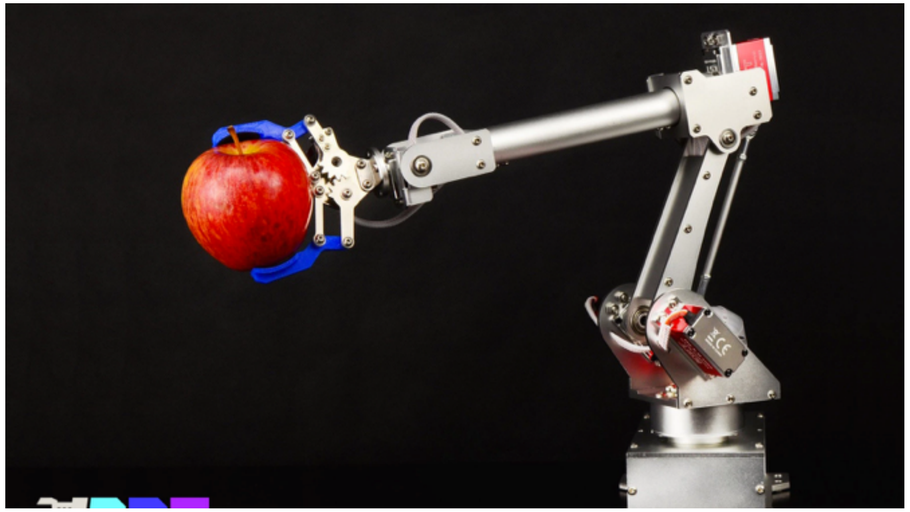
Routers: MLPs
Transformers: combination of convolutions + ReLU + pooling
Note: this work considers Directed Acyclic Graphs.
No architecture learning ✗

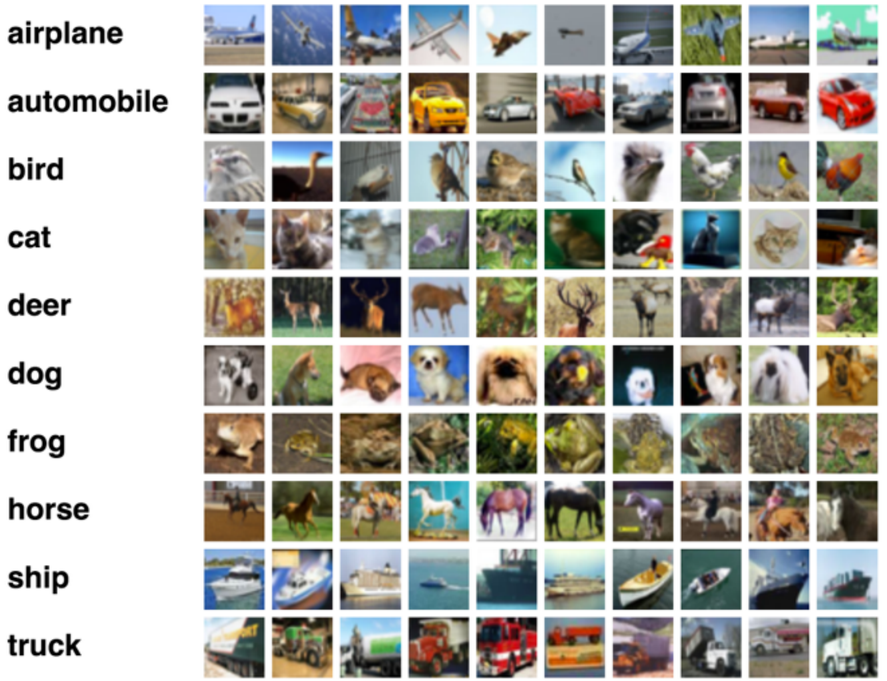
ANTs ✓✓✓

Experiments & Results

Experiments

- **Data:** SARCOS (regression) & MNIST + CIFAR-10 (classification)





Experiments

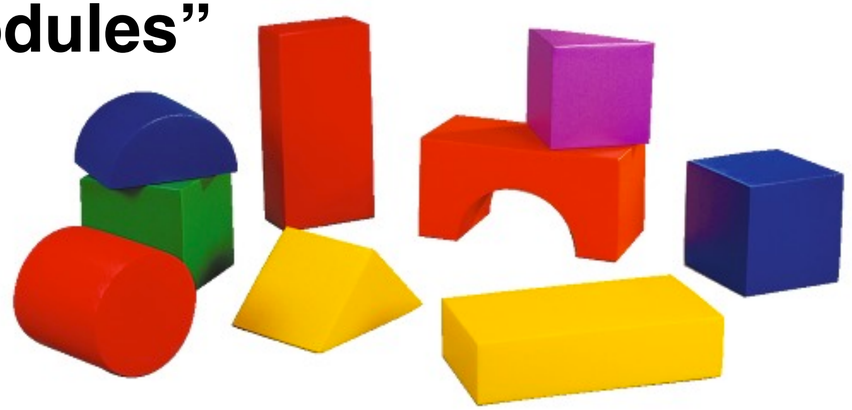
- **Data:** SARCOS (regression) & MNIST + CIFAR-10 (classification)
- **Analysis:**
    1. Comparison with relevant DT and NN models
    2. Learned hierarchical structures
    3. Adaptive model complexity
    4. Effect of global refinement

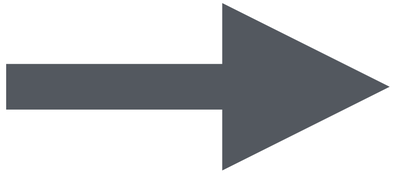
Experiments

- **Models:** choices of *building blocks*

Model	Router, $\mathcal{R}$	Transformer, $\mathcal{T}$	Solver, $\mathcal{S}$	Downsample Freq.
ANT-MNIST-A	$1 \times$ conv5-40 + GAP + 2×FC	$1 \times$ conv5-40	LC	1
ANT-MNIST-B	$1 \times$ conv3-40 + GAP + 2×FC	$1 \times$ conv3-40	LC	2
ANT-MNIST-C	$1 \times$ conv5-5 + GAP + 2×FC	$1 \times$ conv5-5	LC	2
ANT-CIFAR10-A	$2 \times$ conv3-128 + GAP + 1×FC	$2 \times$ conv3-128	LC	1
ANT-CIFAR10-B	$2 \times$ conv3-96 + GAP + 1×FC	$2 \times$ conv3-96	LC	1
ANT-CIFAR10-C	$2 \times$ conv3-72 + GAP + 1×FC	$2 \times$ conv3-72	GAP + LC	1
ANT-SARCOS	$1\times$ MLP ($h = 128$)	$1\times$ FC ($h = 128$) + tanh	LR	0

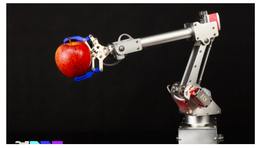
“modules”



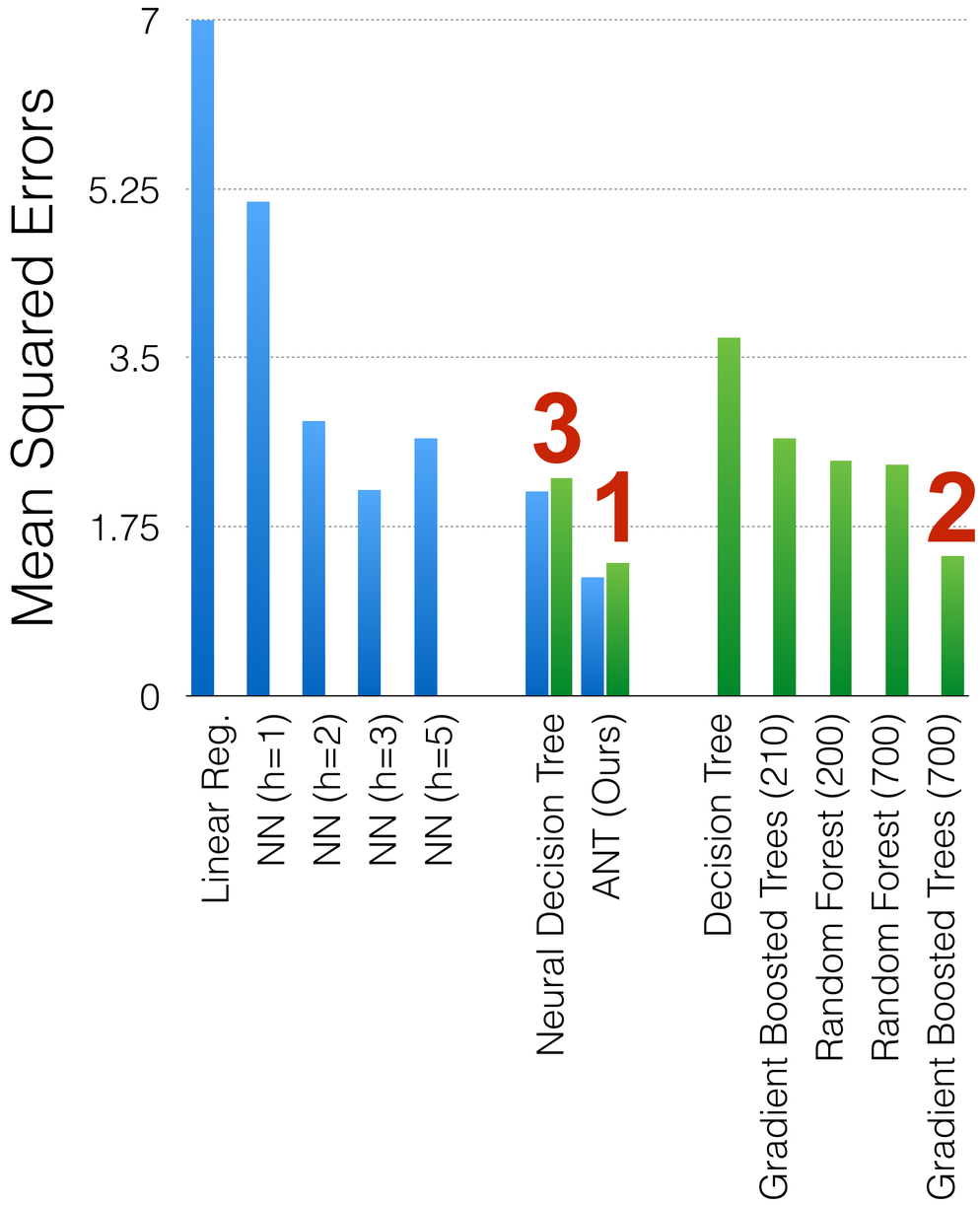


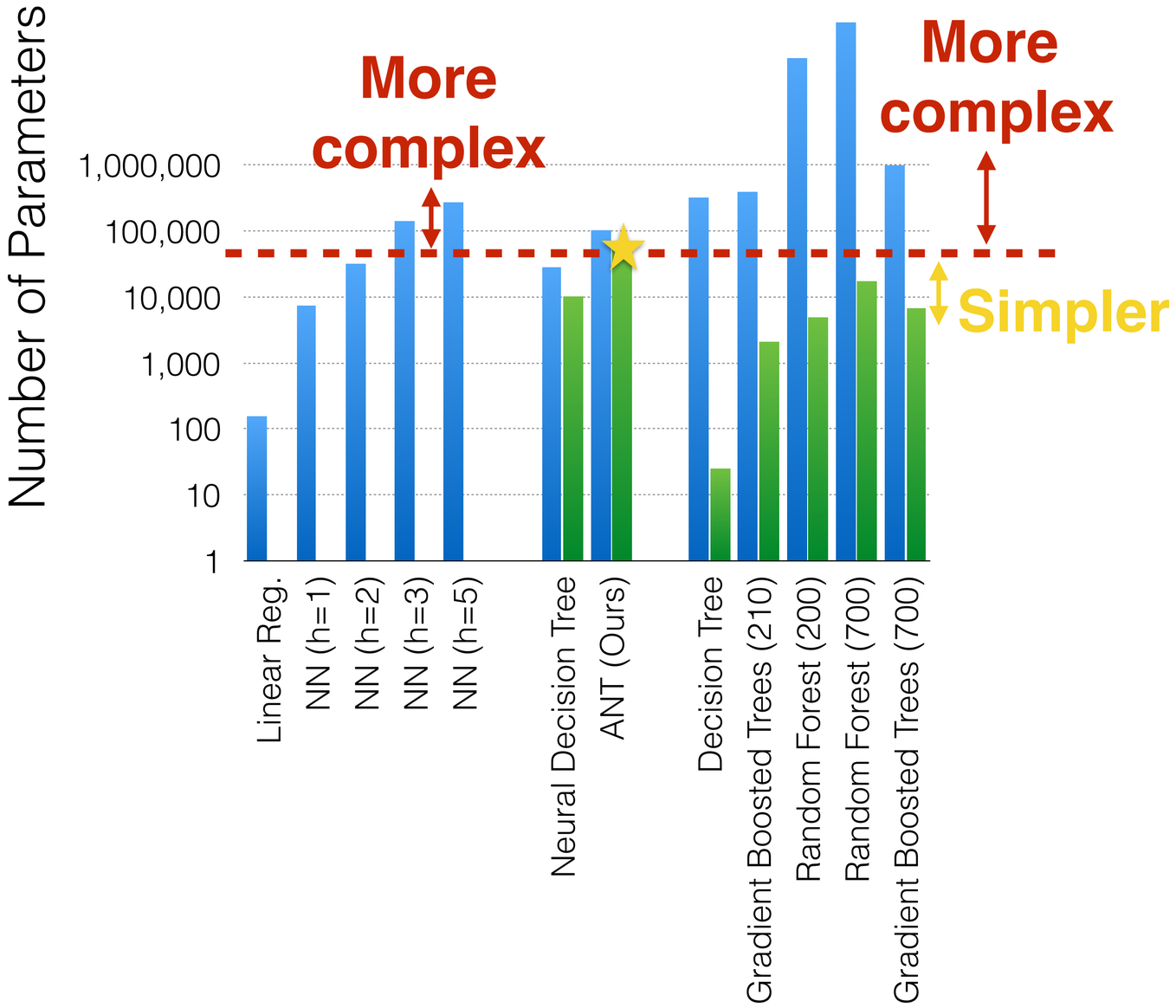
“ANTs”

SARCOS regression



Multi-path inference
Single-path inference



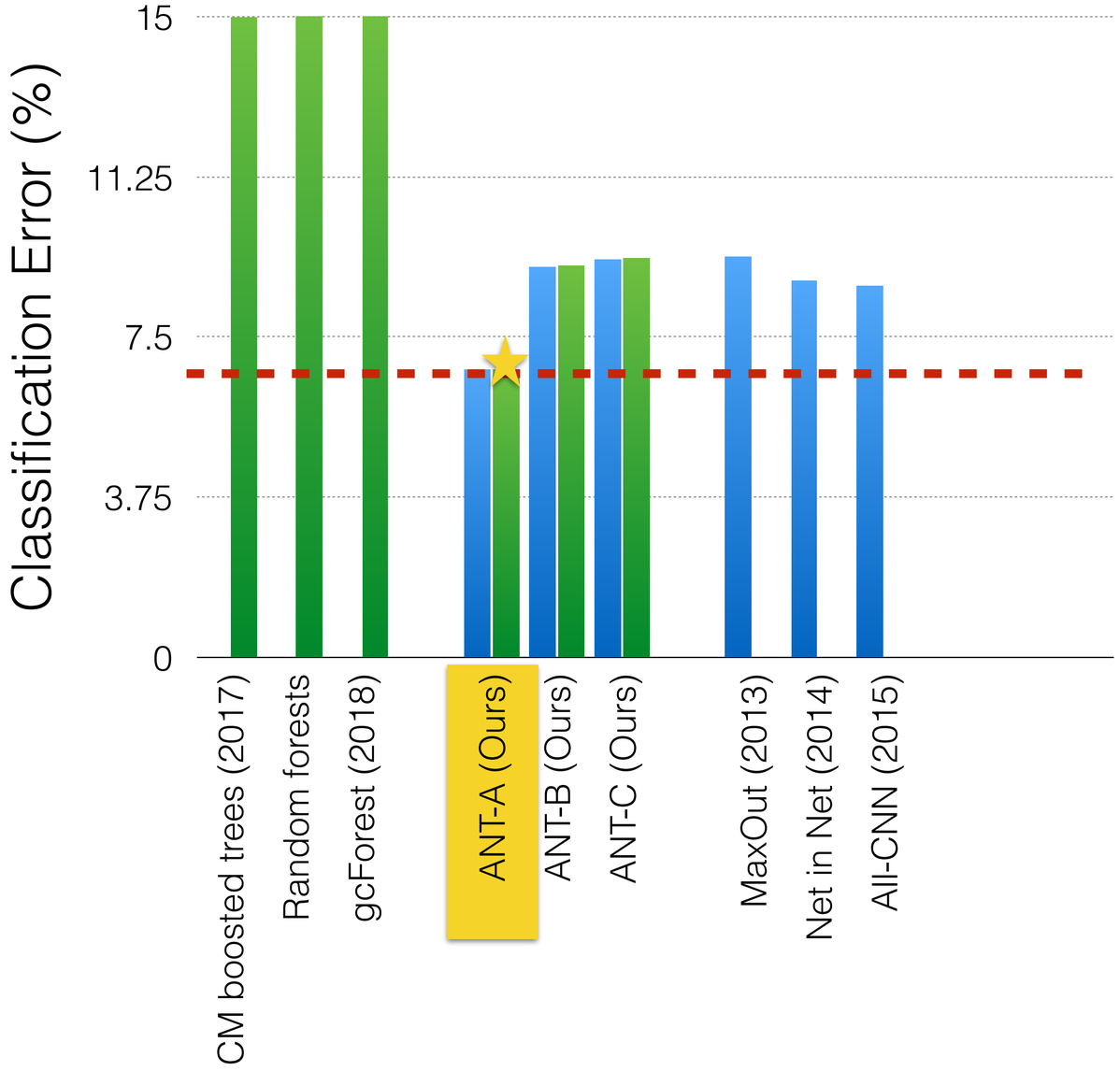


CIFAR-10

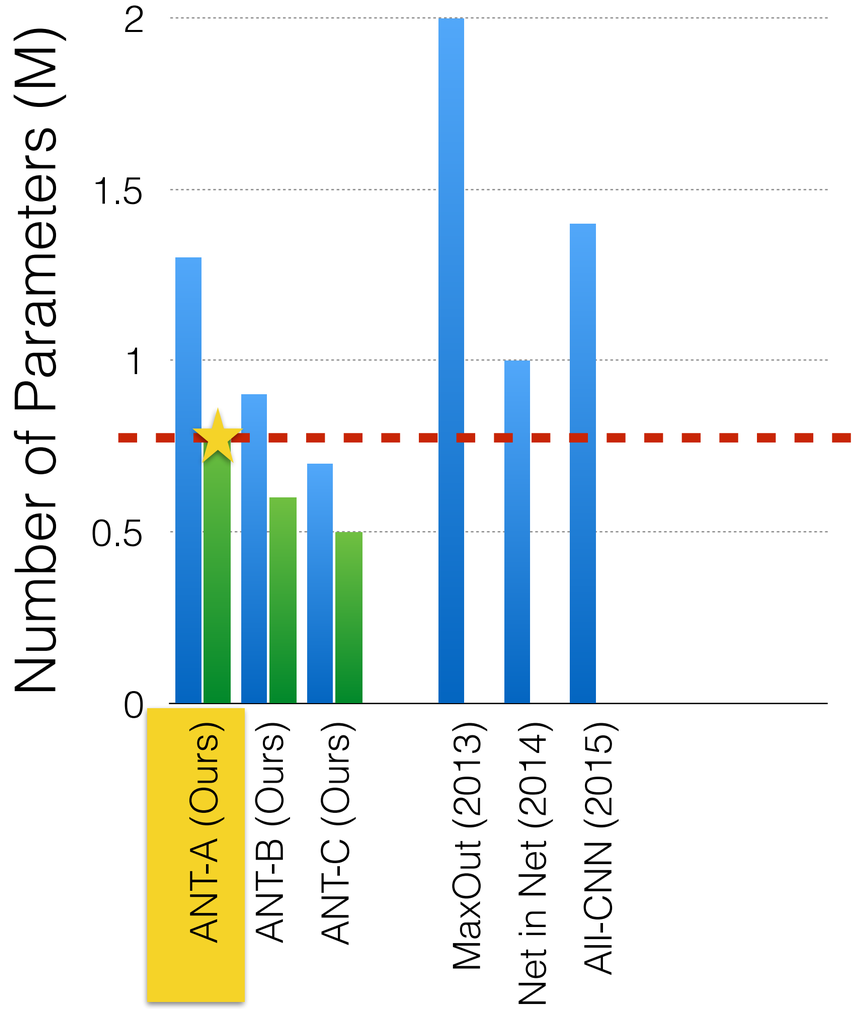
Multi-path inference

Single-path inference

> 25% error



>2M

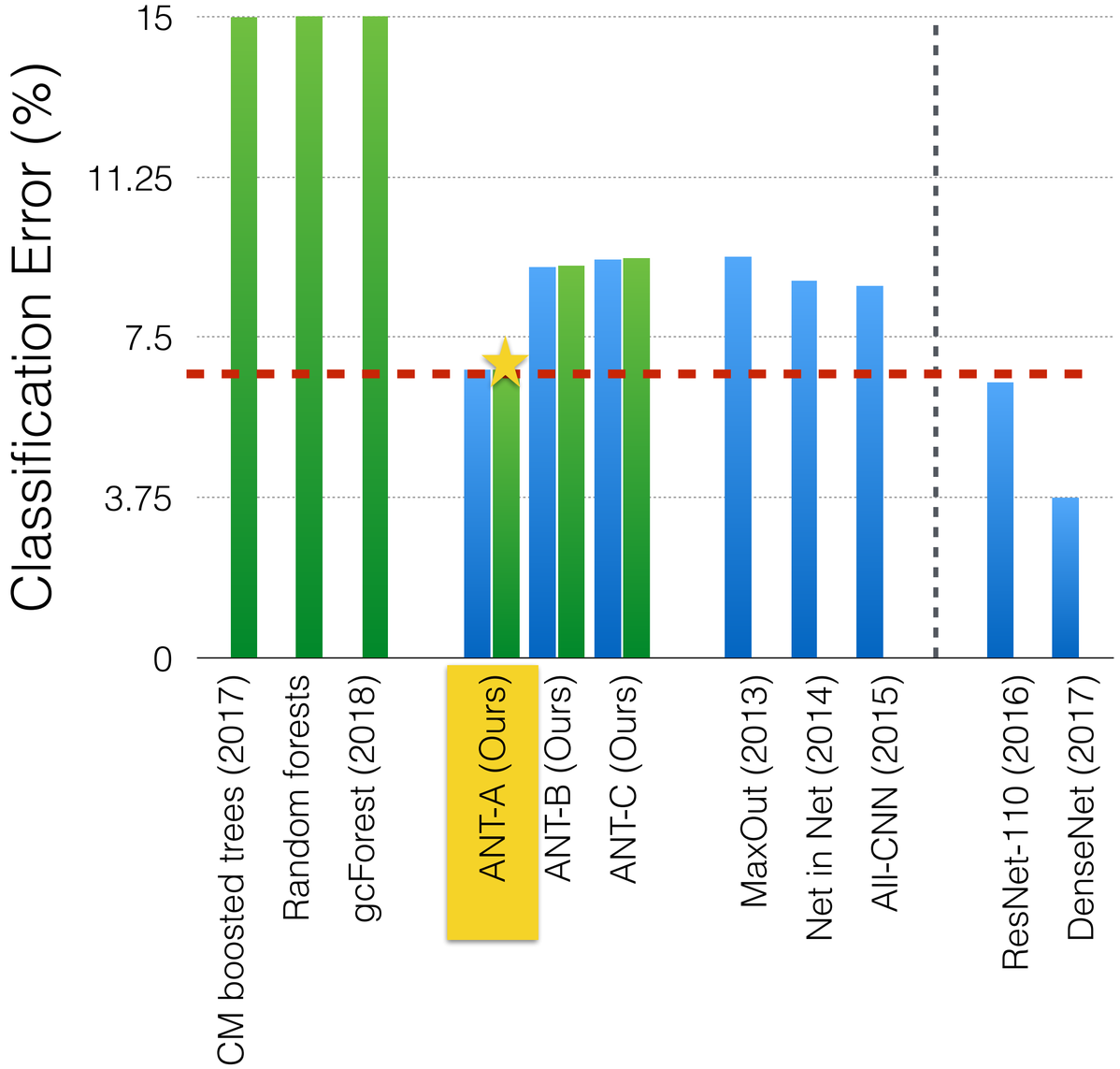


CIFAR-10

Multi-path inference

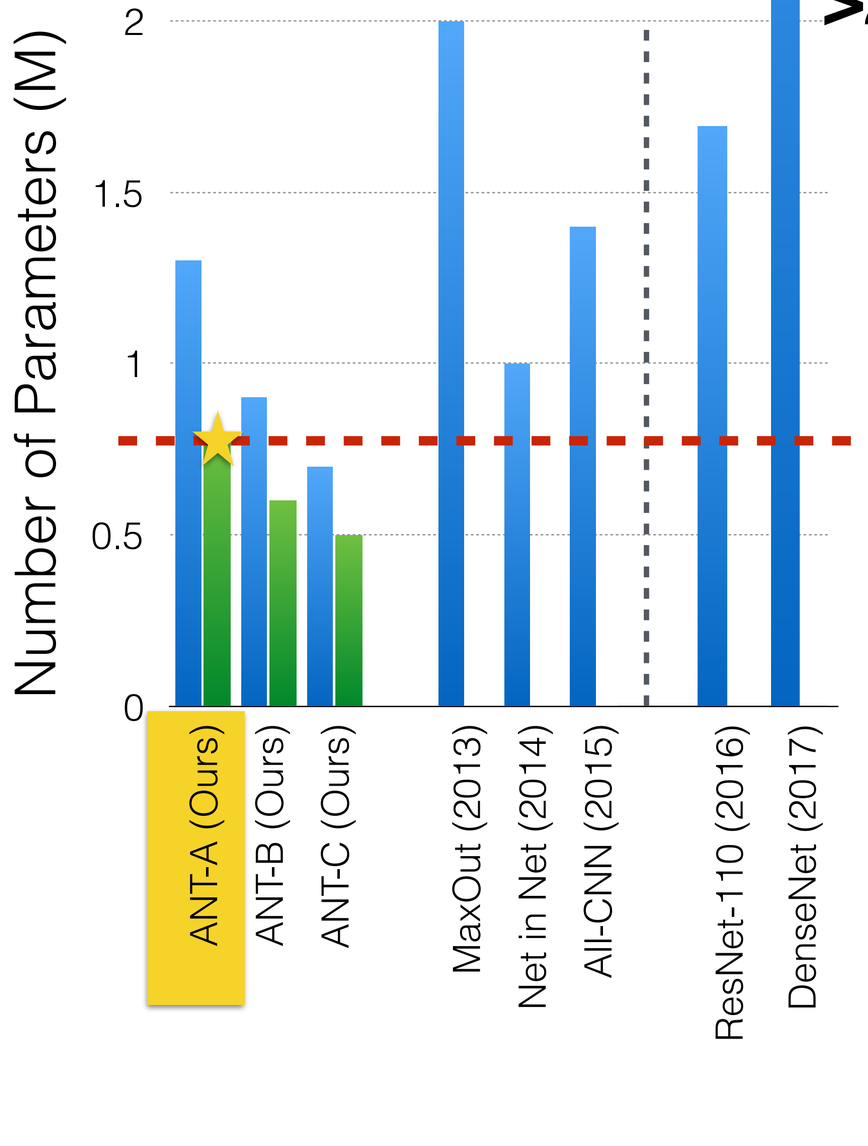
Single-path inference

> 25% error

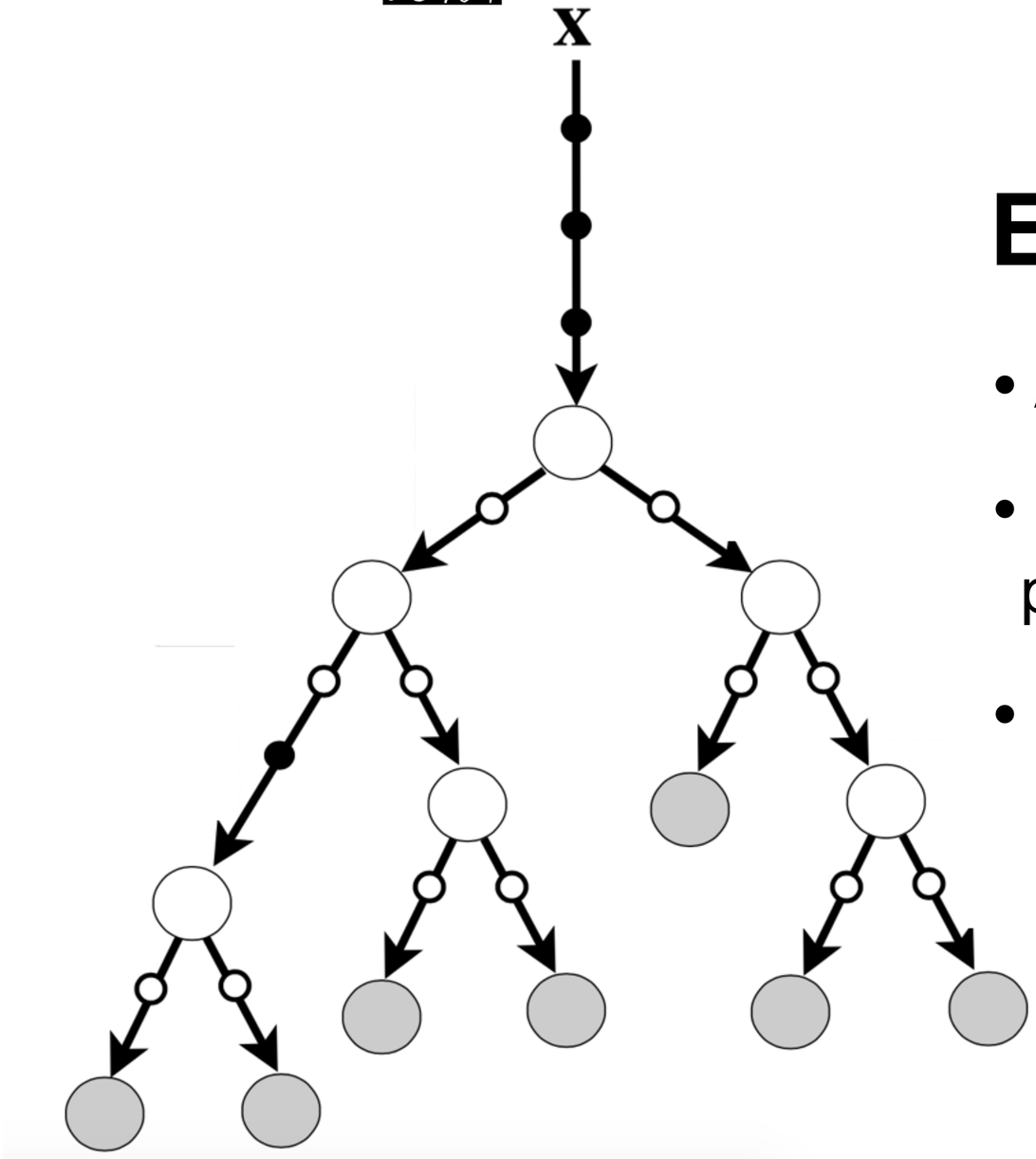


>2M

>25M



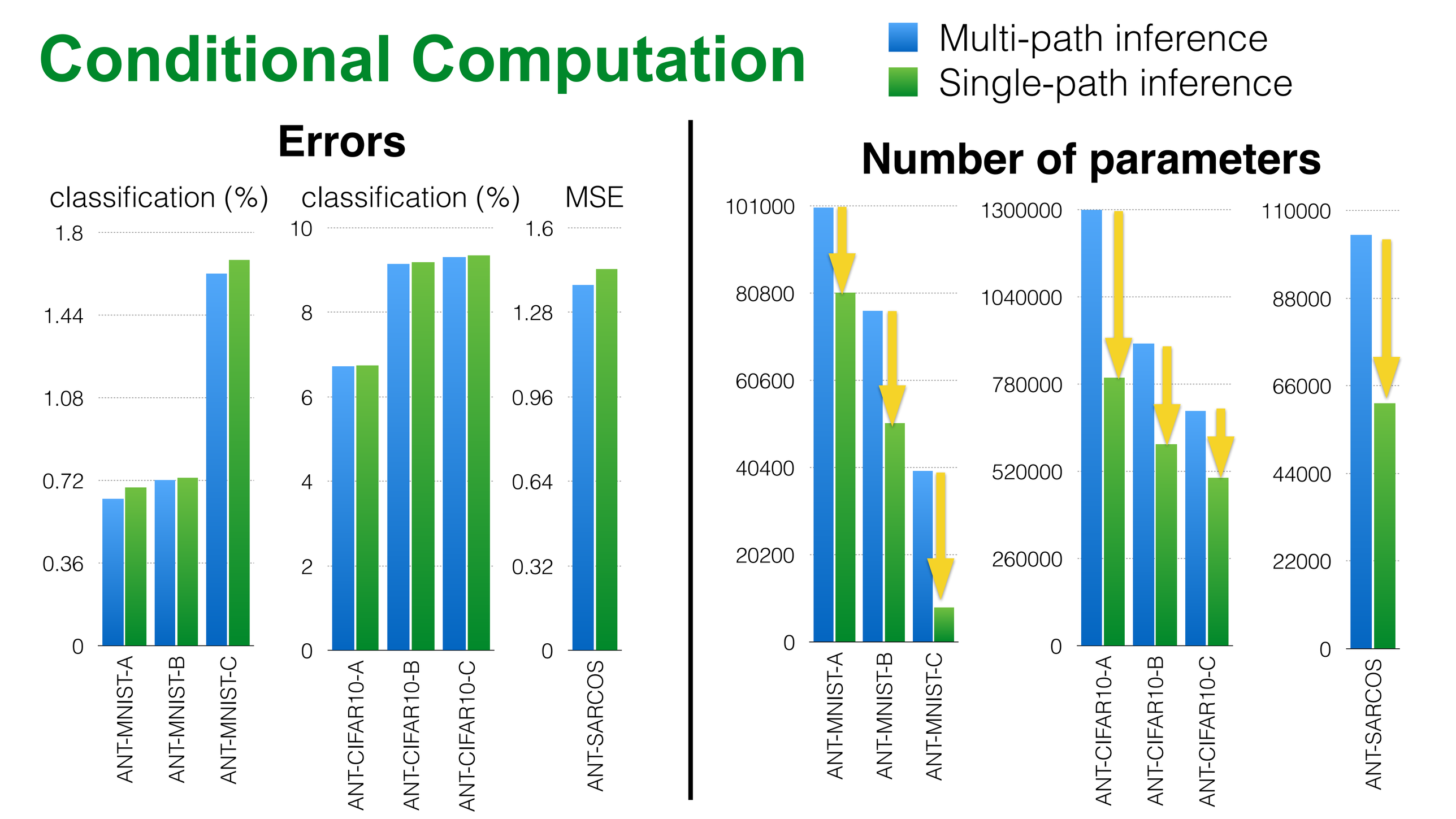
MNIST



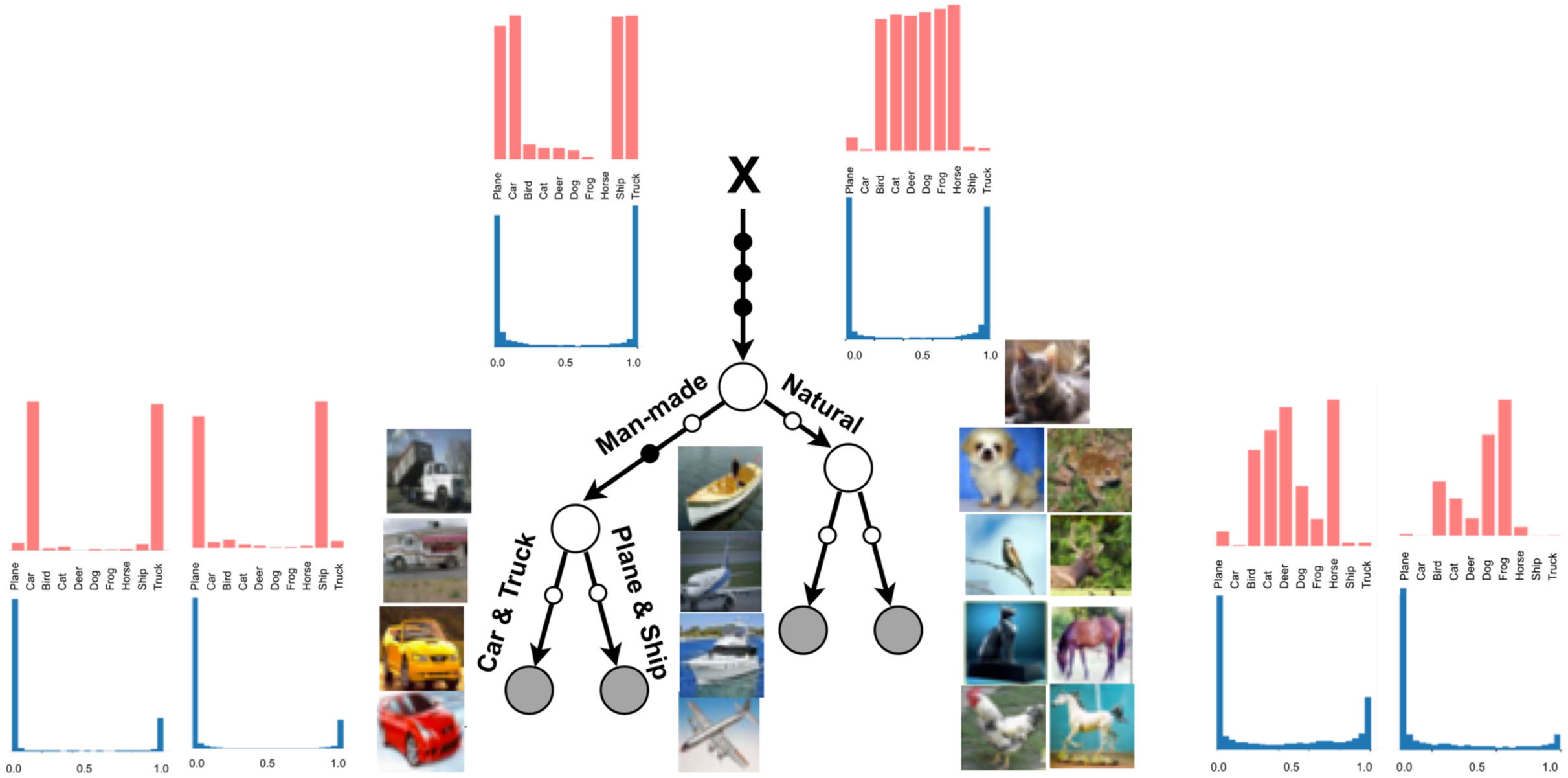
Example: “ANT-MNIST-C”

- Achieves > 98% acc.
- Single-path inference requires as few param. (~ 8000) as a **linear classifier**
- Linear classifier only achieves 92% acc.

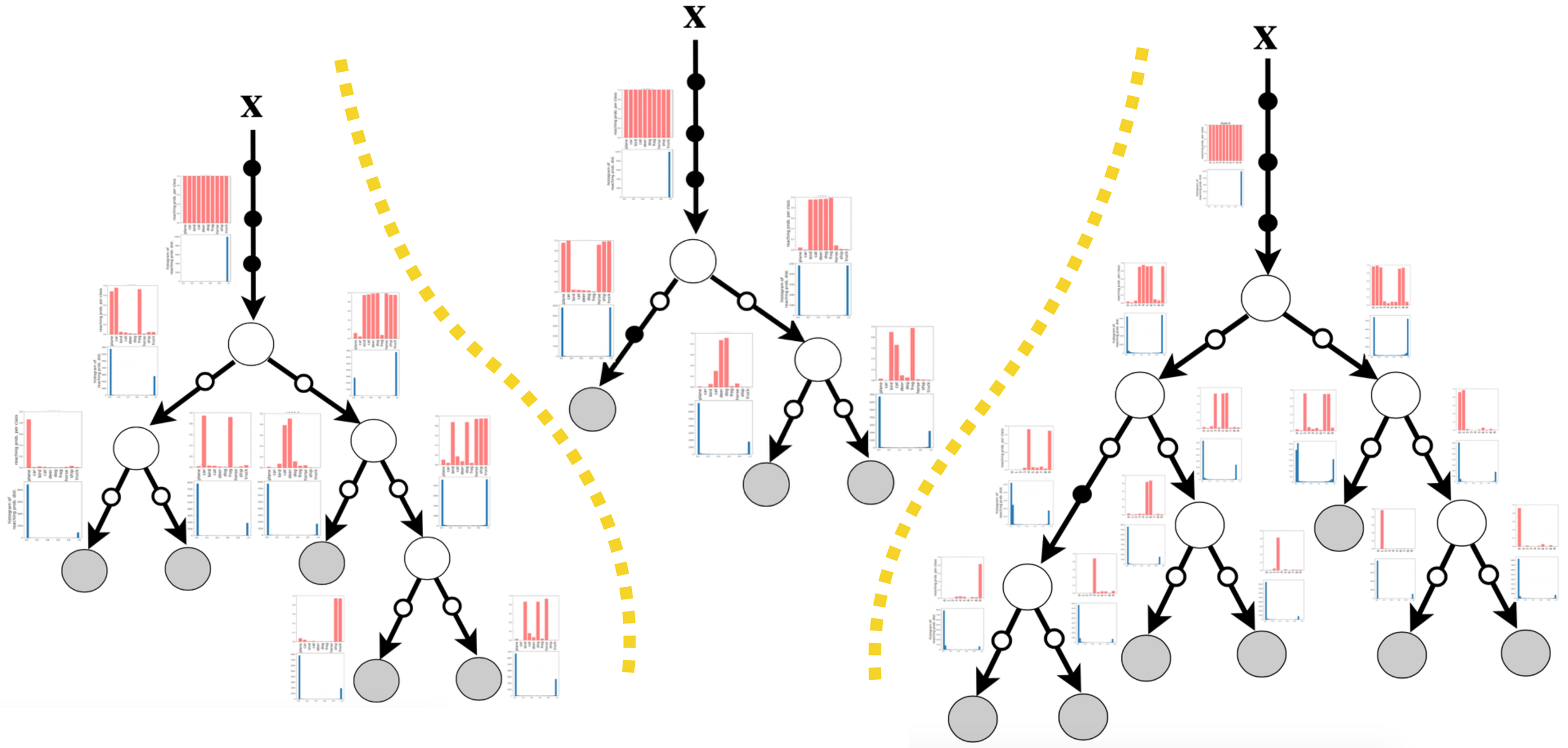
Conditional Computation



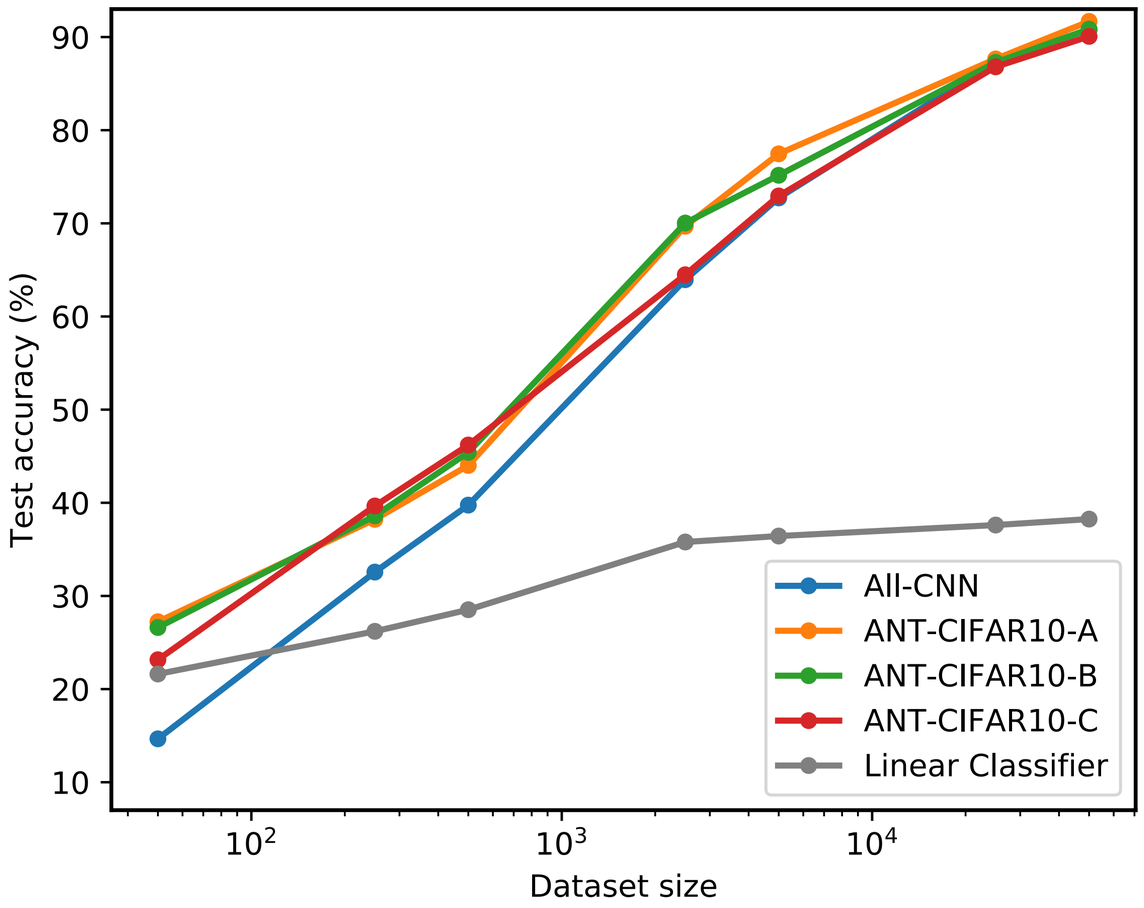
Learned Hierarchy on CIFAR-10

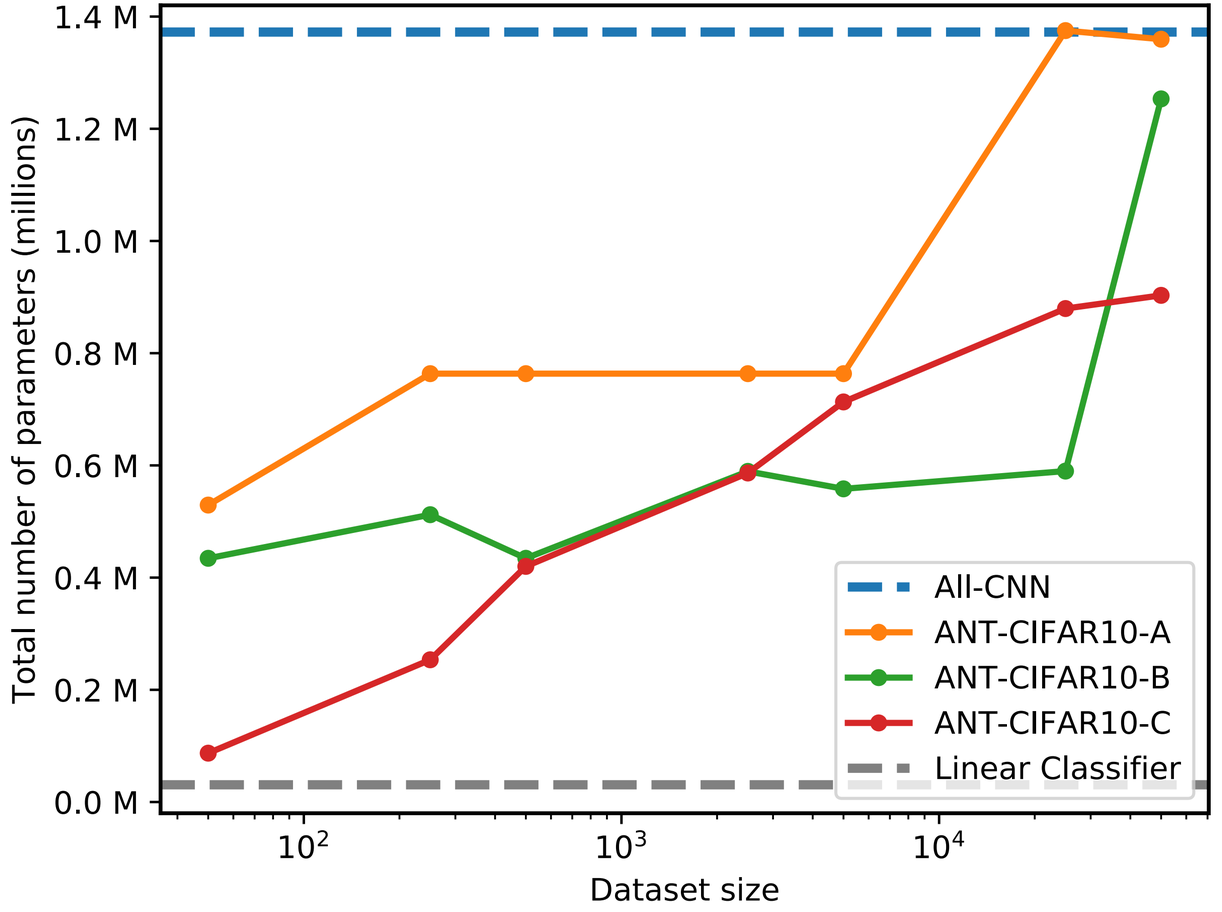


Unbalanced Trees: Adaptive Computation



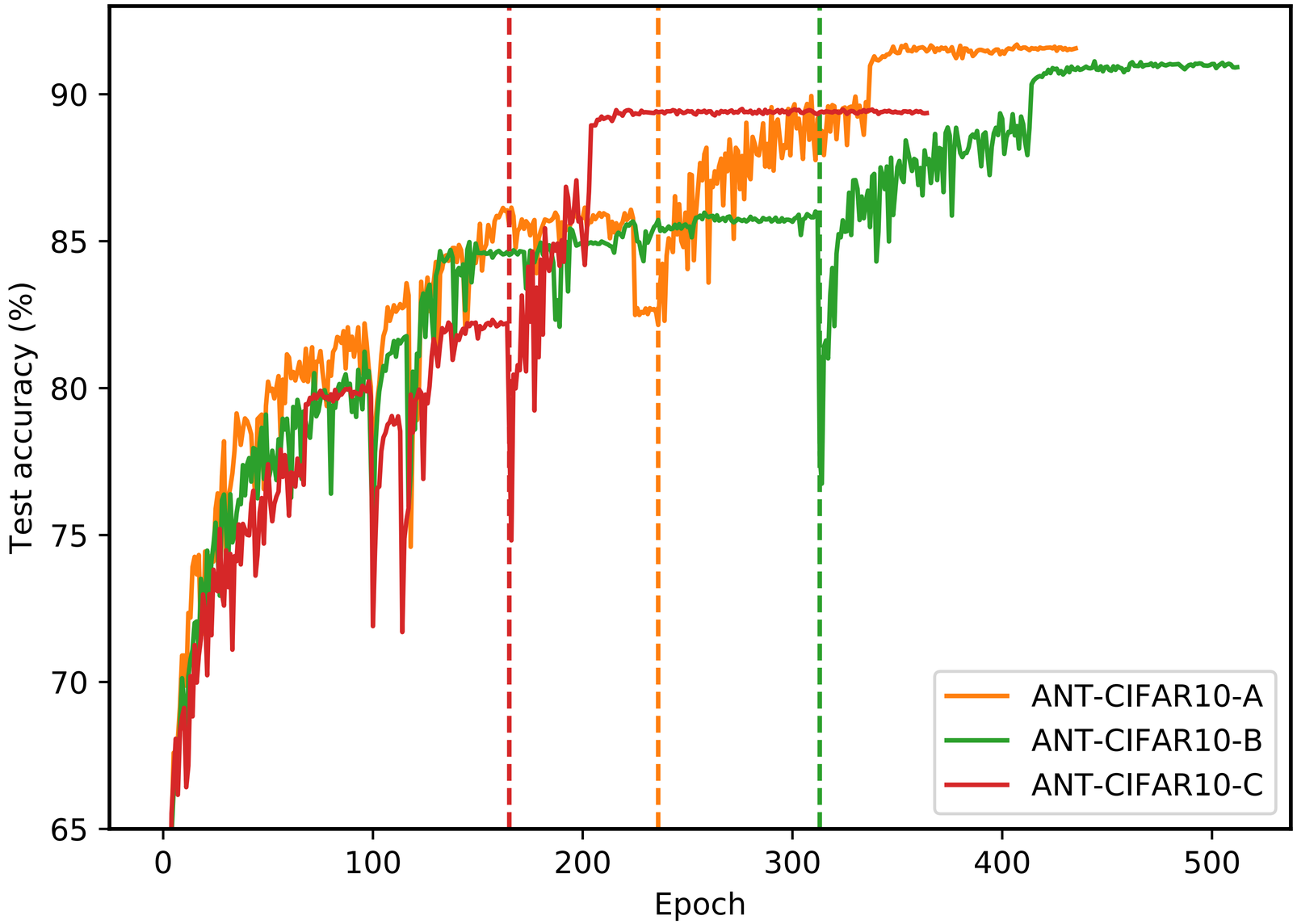
Adaptive Model Complexity



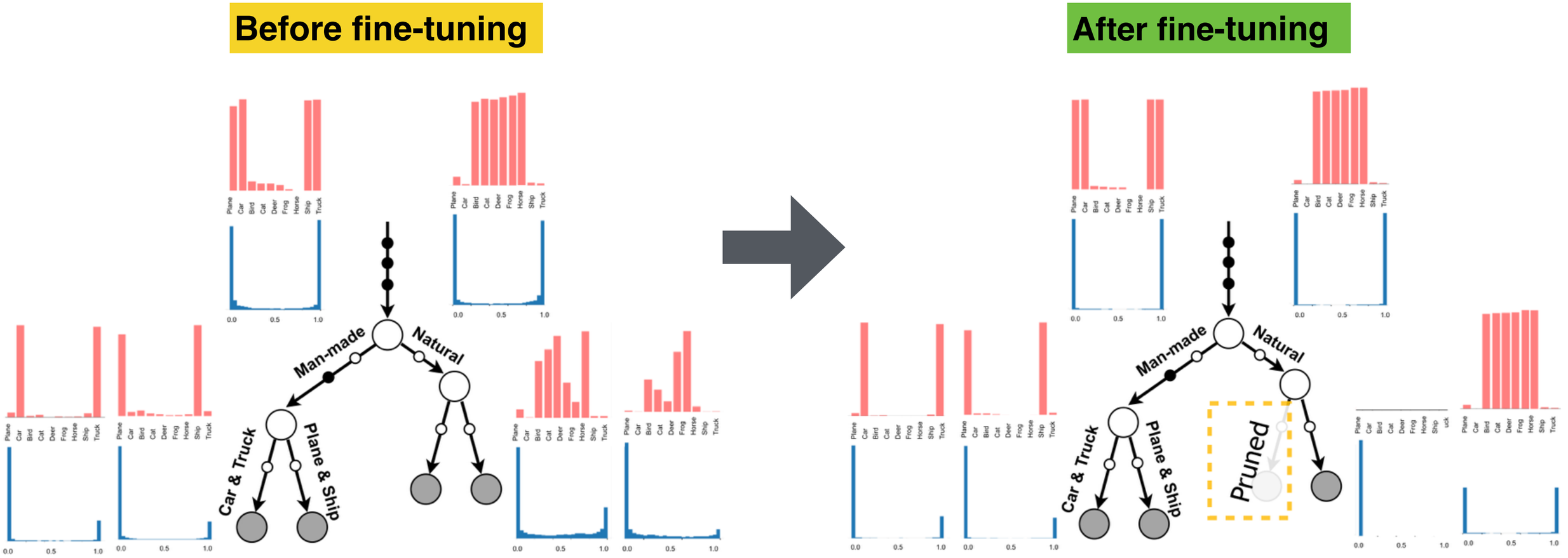


Models are trained on subsets of size 50, 250, 500, 2.5k, 5k, 25k, 45k examples.

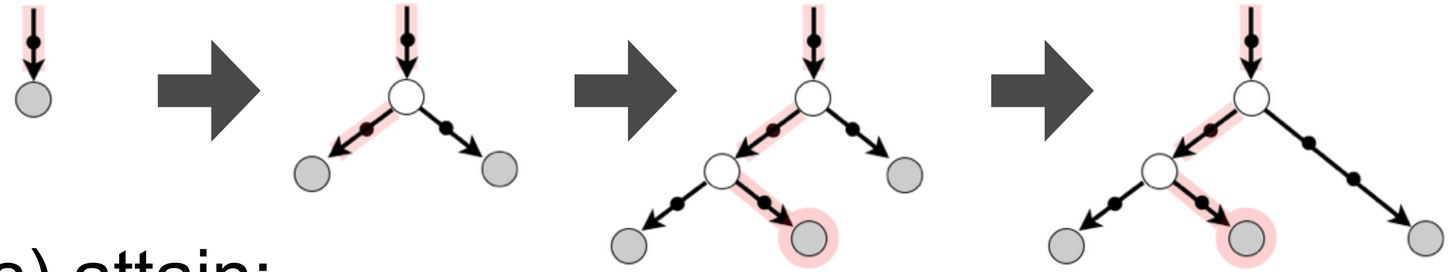
“Pruning” via Global Refinement



“Pruning” via Global Refinement



Summary



Adaptive Neural Trees (ANTs) attain:

- **Freedom to decide when to share & separate representation:**
  - achieves high accuracy with lightweight inference
  - learns hierarchical structures in data useful to the end-task

- **Adaptive model complexity**:
  - grows the architecture, tuning to the size and complexity of data

Future Challenges

- **Scale up to larger, higher dimensional datasets**
  - more *effective* optimisation (non-local?)
  - more *efficient* optimisation (stochastic?)
- **Evaluate “interpretability” (algorithmic transparency)**
- **Extend to other problems:**
  - structured prediction, generative models, continual learning

Acknowledgements

Kai Arulkumaran

Daniel C. Alexander

Antonio Criminisi

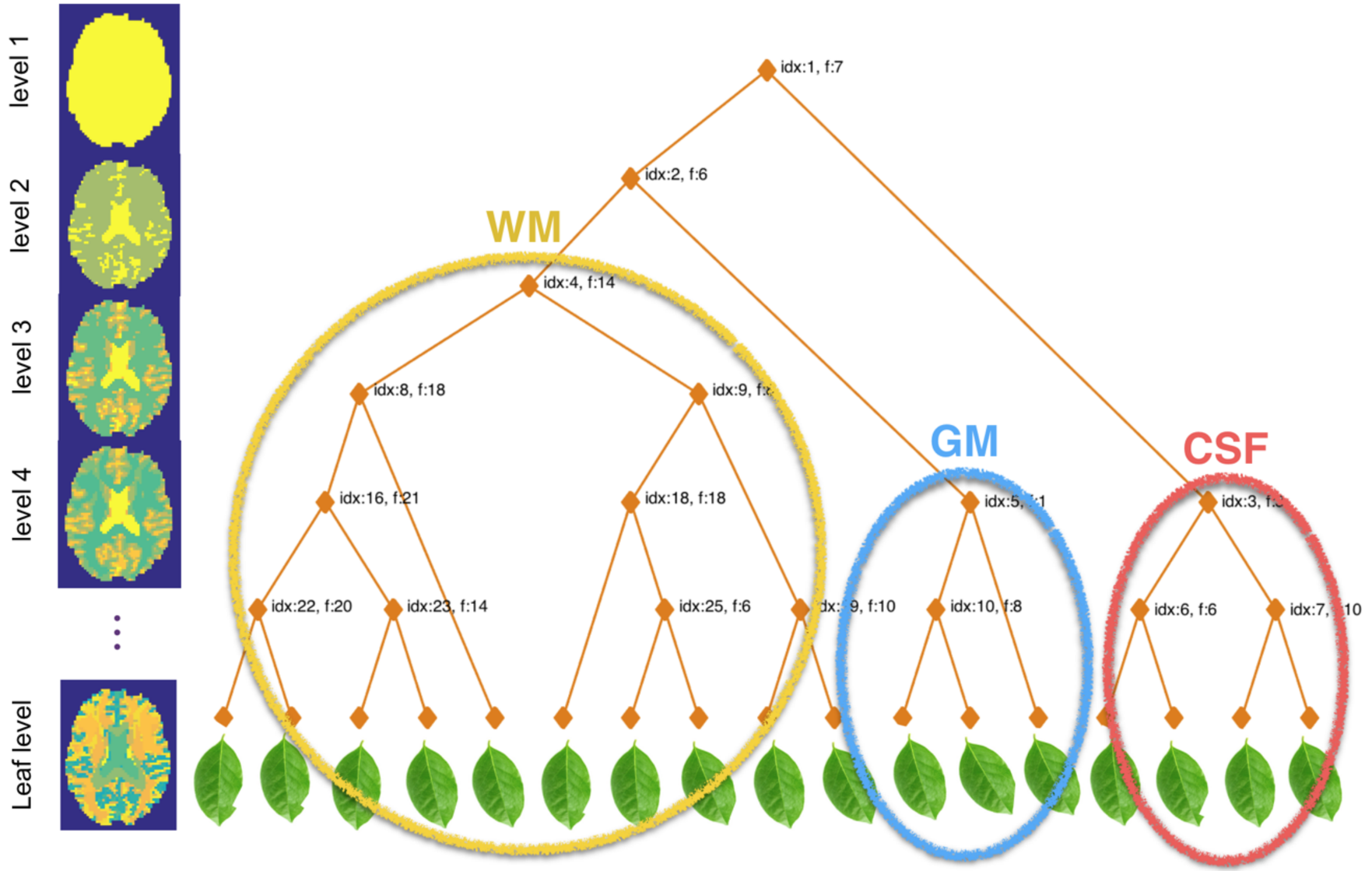
Aditya Nori

I would also like to thank:
Konstantinos Kamnitsas, Jan Stuehmer, Danielle Belgrave, Sebastian Tschiatschek

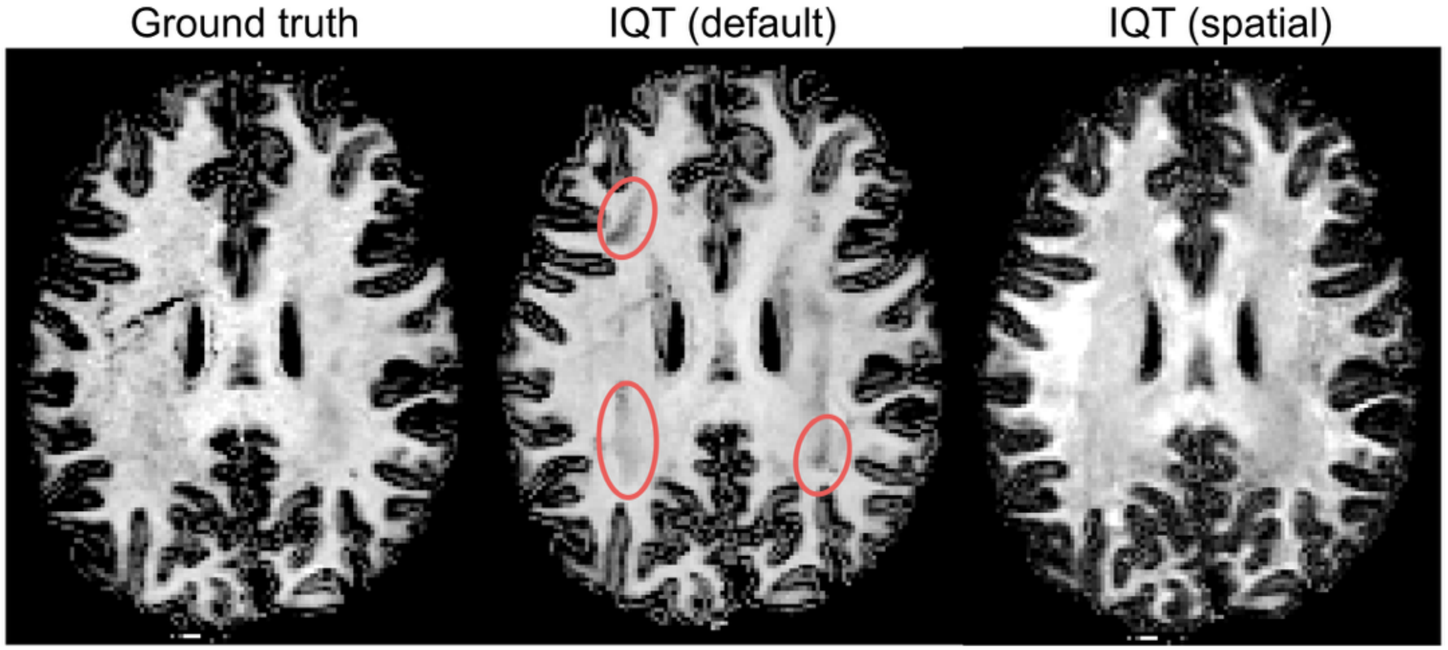
cmic
Centre for Medical Image Computing

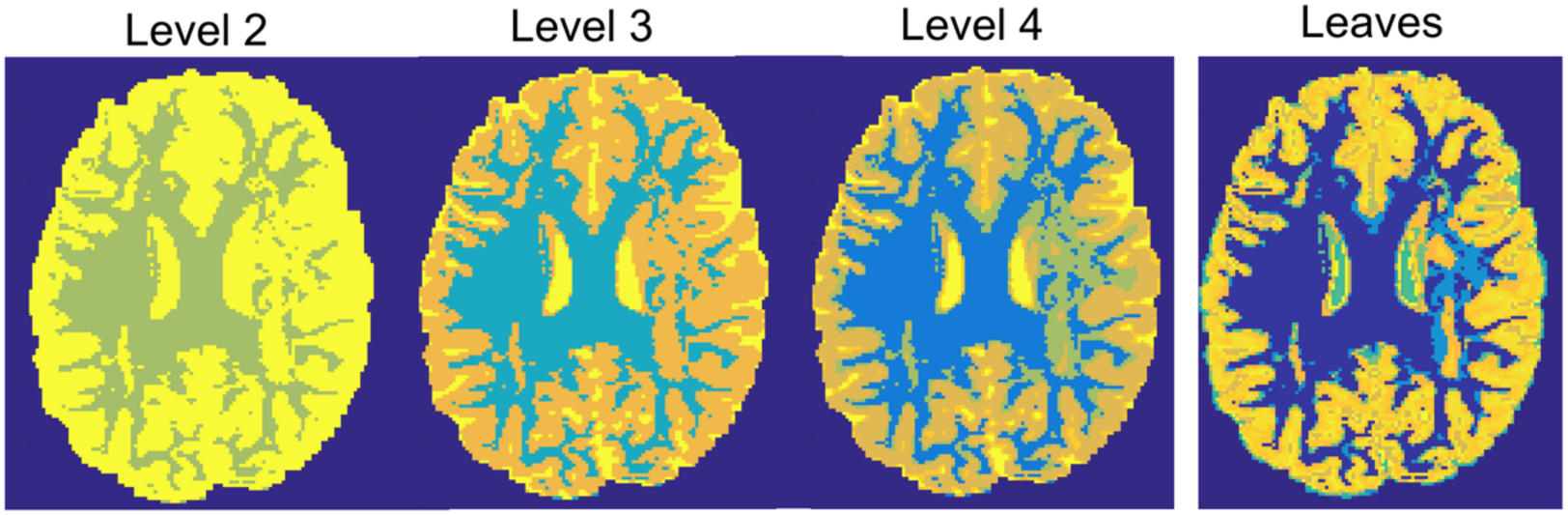
Microsoft Research Cambridge

Debugging a Decision Tree



Debugging a Decision Tree





Training Time

Tab. 5 summarises the time taken on a single Titan X GPU for the growth phase and refinement phase of various ANTs, and compares against the training time of All-CNN (Springenberg et al., 2015). Local optimisation during the growth phase means that the gradient computation is constrained to the newly added component of the graph, allowing us to grow a good candidate model under 2 hours on a single GPU.

Table 5: Training time comparison. Time and number of epochs taken for the growth and refinement phase are shown. along with the time required to train the baseline, All-CNN (Springenberg et al., 2015).

	Growth		**Fine-tune**	
Model	Time	Epochs	Time	Epochs
All-CNN (baseline)	–	–	1.1 (hr)	200
ANT-CIFAR10-A	1.3 (hr)	236	1.5 (hr)	200
ANT-CIFAR10-B	0.8 (hr)	313	0.9 (hr)	200
ANT-CIFAR10-C	0.7 (hr)	285	0.8 (hr)	200

Benefits of ensembling

Table 9: Comparison of prediction errors of a single ANT versus an ensemble of 8, with predictions averaged over all ANTs in the ensemble.

	MNIST (Class Error %)		**CIFAR-10** (Class Error %)		**SARCOS** (MSE)	
	Error (Full)	Error (Path)	Error (Full)	Error (Path)	Error (Full)	Error (Path)
Single model	0.64	0.69	8.31	8.32	1.384	1.542
Ensemble	0.29	0.30	7.76	7.79	1.226	1.372

Table 10: Parameter counts for a single ANT versus an ensemble of 8.

	MNIST		**CIFAR-10**		**SARCOS**	
	Params. (Full)	Params. (Path)	Params. (Full)	Params. (Path)	Params. (Full)	Params. (Path)
Single model	100,596	84,935	1.4M	1.0M	103,823	61,640
Ensemble	850,775	655,449	8.7M	7.4M	598,280	360,766

Ablation Study

Module Spec.	Error % (Full)			Error % (Path)		
	ANT (default)	CNN (no routers)	SDT/HME (no transformers)	ANT (default)	CNN (no routers)	STD/HME (no transformers)
ANT-MNIST-A	0.64	0.74	3.18	0.69	0.74	4.19
ANT-MNIST-B	0.72	0.80	4.63	0.73	0.80	3.62
ANT-MNIST-C	1.62	3.71	5.70	1.68	3.71	6.96
ANT-CIFAR10-A	8.31	9.29	39.29	8.32	9.29	40.33
ANT-CIFAR10-B	9.15	11.08	43.09	9.18	11.08	44.25
ANT-CIFAR10-C	9.31	11.61	48.59	9.34	11.61	50.02

Module Spec.	Error (Full)			Error (Path)		
	ANT (default)	NN (no routers)	SDT/HME (no transformers)	ANT (default)	NN (no routers)	SDT/HME (no transformers)
ANT-SARCOS	1.384	2.511	2.118	1.542	2.511	2.246

- No routers = standard CNNs
- No transformers = SDTs / HMEs
- Ablation of transformers/routers leads to higher classification errors.

Learned Hierarchical Structures

Module Spec.	Error % (Selected path)	Error % (Least likely path)
ANT-MNIST-A	0.69	86.18
ANT-MNIST-B	0.73	81.98
ANT-MNIST-C	1.68	98.84
ANT-CIFAR10-A	8.32	74.28
ANT-CIFAR10-B	9.18	89.74
ANT-CIFAR10-C	9.34	97.52